Enhanced low-power high-speed probabilistic adders for error-tolerant application

Zhu, Ning

2011

Zhu, N. (2011). Enhanced low-power high-speed probabilistic adders for error-tolerant application. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46273>

<https://doi.org/10.32657/10356/46273>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**ENHANCED LOW-POWER HIGH-
SPEED PROBABILISTIC ADDERS FOR
ERROR-TOLERANT APPLICATION**

ZHU NING

**SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING**

2011

ENHANCED LOW-POWER HIGH- SPEED PROBABILISTIC ADDERS FOR ERROR-TOLERANT APPLICATION

ZHU NING

School of Electrical and Electronics Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of

Master of Engineering

2011

Acknowledgement

Firstly, I would like to express my most sincere gratitude to my supervisors, Associate Professor Goh Wang Ling, for her countless help and continuous supports throughout the project. Her knowledgeable advices and guidance are indispensable for the completion of this project. The knowledge and thoughts I have gained from them through the numerous discussions with them will definitely benefit my future life.

I also would like to thank Professor Yeo Kiat Seng and Dr. Kong Zhi Hui, for their help in the project. Their suggestions from them and the discussions with them give me a lot of ideas and help me a lot for the research.

In addition, I like to thank Mr. Zhang Wei Jia and Mr. Wang Gang, for their kind help in the course of the project. The discussions with them kindled my thought. They are also the co-authors of my two published/submitted papers, respectively.

Last but not least, I would like to thank Nanyang Technological University for providing me the research scholarship to support me to complete the project.

Abstract

In modern VLSI technology, the occurrence of all kinds of errors has become inevitable. To overcome all the possible errors is a very expensive task. It not only consumes a lot of power but also degrades the speed performance. With the concept of “error-tolerant” (ET), which allows the existence of certain amount of errors, improvements in power consumption and/or other performance metrics can be achieved. In this design, the concept of error-tolerant has been extended to the field of circuit design. When “imperfect” algorithms and circuit structures are employed, a substantial yield for an error-tolerant digital circuit, in terms of power consumption, speed performance, and transistor count, can be realized.

An adder is the basic digital circuit component, which is widely used in many areas. Adopting the ideas and techniques in Error-Tolerant technology in the design of digital adders, a innovative type of adder—Probabilistic Adder for Error-Tolerant Application (ETA) has been designed. In conventional design, obtaining high speed usually means more power will be consumed and low power will normally degrade the speed of a circuit. To breakthrough this bottleneck in conventional technologies for designing a truly low-power and high-speed digital circuit, a new metric besides **power** and **speed** should be brought into the design process. In the proposed designs, **accuracy** plays the role of such a new metric. By sacrificing some degree of accuracy, great improvements in both power consumption and speed performance can be achieved.

Several different implementations of the enhanced ETA have been proposed in the report, namely the ETA Type II (ETAII), ETA Type III (ETAIII) and ETA Type IV (ETAIV).

The ETAII is based on the idea that in most cases the carry signal for a bit position is determined by several neighboring bits instead of all the bits on its right. Hence, the critical path of the whole circuit can be greatly curtailed by dividing the whole adder into a number of blocks and conducting the addition operations in each block concurrently. With this method, improvements in power consumption and speed performance can be achieved. Simulation results show that the Power-Delay Product

(PDP) of ETAlIM (a modified structure of the original ETAlI) is 60%, 75%, 81%, and 71% better than that of the RCA, CSK, CSL, and CLA, respectively.

The basic idea of ETAlII is to combine the advantages of ETAl and ETAlI, it adopts both the idea of “accurate part – inaccurate part” mode of ETAl and the “block separation mode” in ETAlI, this architecture makes ETAlII can have better accuracy for all ranges of the inputs (the Acceptable Probability (AP) is evenly distributed in different range of inputs but not only good for large inputs / small inputs). Simulation results show that the ETAlII’s PDP is also far better than the conventional adders and remains a same level with ETAl and ETAlI. However, its accuracy is much better than both ETAl and ETAlI, with the trade off that it has a large area (transistor count) for the duplicate FAs/SAs in the inaccurate part.

ETAlV is based on the concept of “block dividing” as in ETAlI, but different with ETAlI, it adopts the architecture of “Carry-Select-Adder” instead of “Carry-Lookahead-Adder”. The 2-to-1 multiplexer used in the circuit will help to get better accuracy while remaining a fast speed. The simulation results show that the ETAlV still remains a very small PDP as the same level of ETAl series which is far better than the conventional adders. Although its area is a bit larger than the previous ETAs, its accuracy is better.

The applications of the error-tolerant circuit/system also be studied and investigated, such as sound application, image application etc. All the applications related to the human sense can adopt the error-tolerant concept. And the concept of ETAl has been proven to be feasible by employing it in digital image processing. It is therefore logical to extend this concept to other digital circuits such as the multiplier, and to a whole DSP system.

Glossary

Term	Definition
ETA	Probabilistic Adder for Error Tolerant Application
DSP	Digital Signal Processing
VLSI	Very-Large-Scale Integration
DTAD	Digital Telephone Answering Device
ADC	Analog to Digital Converter
TGA	Transmission Gate Adder
RCA	Ripple Carry Adder
LSB	Least Significant Bit
MSB	Most Significant Bit
CSK	Carry Skip Adder
CSL	Carry Select Adder
CLA	Carry Lookahead Adder
ESCT	Estimate Carry Structure
OE	Overall Error
ACC	Accuracy
MAA	Minimum Acceptable Accuracy
AP	Acceptance Probability
PDP	Power-Delay Product
TC	Transistor Count
FA	Full Adder
SA	Speed Adder
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
PSD	Power Spectral Density
FPGA	Field Programmable Gate Array

Table of Contents

Chapter 1: Introduction	9
1.1 Background	9
1.2 Motivation	9
1.3 Objective	11
1.4 Contribution of the thesis	11
1.5 Organization of the Thesis	12
Chapter 2: Literature review	13
2.1 Error-Tolerance	13
2.1.1 Concepts	13
2.1.2 Integrated Circuit Testing Methodology that Support Error-Tolerance	14
2.1.3 A Case Study of Error-Tolerance	15
2.2 Conventional Designs of Digital Adders	17
2.2.1 Half adder and Full adder	17
2.2.2 Ripple-Carry Adder	21
2.2.3 Carry-Skip Adder	21
2.2.4 Carry-Select Adder	22
2.2.5 Carry-Lookahead Adder	24
2.3 Existing Probabilistic/truncated system	27
2.3.1 Carry State Estimation for Adders	27
2.3.2 Probabilistic Multiple Carry Estimates	29
2.3.3 Probabilistic Ripple-Carry Adder	30
2.4 Summary	32
Chapter 3: Probabilistic Adders	33
3.1 Define of terms	33
3.2 Previous work: ETA Type I	34
3.2.1 Addition Algorithm	34
3.2.2 Hardware Implementation	36
3.2.3 Simulation Results	37
3.2.4 Problems of ETA I	38
3.3 ETA Type II	40
3.3.1 Theoretical Analysis	40

3.3.2	Architecture of ETAll-----	41
3.3.3	Dividing Strategy-----	42
3.3.4	Implementation of a 32-bit ETAll-----	43
3.3.5	Accuracy of ETAll for small number inputs-----	44
3.3.6	Modified ETAll-----	45
3.3.7	Comparison with conventional adders-----	46
3.4	ETA Type III -----	47
3.4.1	Architecture of ETAllI -----	48
3.4.2	Hardware Implementation of a 32-bit ETAllI-----	50
3.4.3	Accuracy of ETAllI -----	52
3.4.4	Simulation results of ETAllI -----	52
3.5	ETA Type IV-----	53
3.5.1	Addition Arithmetic for ETAIV -----	53
3.5.2	Dividing strategy for ETAIV -----	55
3.5.3	Implementation of a 32-bit ETAIV -----	56
3.5.4	Accuracy of ETAIV-----	57
3.5.5	Simulation Results of ETAIV -----	58
3.6	Summary-----	59
Chapter 4:	Applications of Probabilistic Adder Circuits-----	61
4.1	Digital Sound Processing-----	61
4.2	Digital Image Processing -----	66
4.3	Summary-----	71
Chapter 5:	Conclusion and Future work -----	72
5.1	Conclusion-----	72
5.2	Future Work-----	73
List of Publications	-----	75
Reference	-----	76
Appendix	-----	80
Appendix A	– C Code of calculation AP for ETAll-----	80
Appendix B	– HSPICE Code of ETAllI-----	83
Appendix C	– HSPICE Code of ETAllI-----	86
Appendix D	– HSPICE Code of ETAIV -----	91
Appendix E	– MATLAB code of FFT -----	95

Appendix F – MATLAB code of ETAll addition algorithm-----	97
---	----

Chapter 1: Introduction

1.1 Background

The famous Moore's Law provides us an important trend in the development of integrated circuit technology. According to Moore's Law, the number of transistors that can be inexpensively placed on an integrated circuit doubles every two years [1]. The trend of Moore's Law has continued for about half a century and it is not expected to stop for another decade. However, as the feature size of the complementary metal-oxide-semiconductor (CMOS) devices approaches the deep sub-micron "nano-scale", significant challenges to sustaining Moore's Law have emerged. Two of these challenges are the impact of noise [2, 3, 4] and achieving low-power consumption [5, 6]. The conventional view towards the unexpected noise is treating it as an impediment and trying the best to eliminate its impact. It is stated in the 2003 International Technology Roadmap for Semiconductors (ITRS) [7] that the increasing noise sensitivity has become an important issue in the design of devices, circuits, and systems due to a reduction in operating voltage by 20% per technology node. However, the requirement for increasing noise immunity contracts with the traditional methodology to achieve low-power consumption, which is addressed by voltage scaling, as reducing the voltage level may greatly degrade the noise immunity of the circuits.

1.2 Motivation

Since occurrence of all kinds of errors has become inevitable, overcoming all the possible errors is a very expensive task. It not only consumes a lot of power but degrades the speed performance. With the concept of "error-tolerant" [8], which allows the existence of certain amount of errors, improvements in power consumption and/or other performance metrics can be achieved.

Since the original concept of “error-tolerant” proposed by Professor Breuer is derived from the perspective of digital integrated circuit IC testing, it concentrates mainly on defect models such as the stuck-at, bridging, and delay faults [8]. The benefits of an error-tolerant circuit are also limited to the cost of manufacturing, verification, and testing. In this report, the concept of “error-tolerant” has been extended from the field of circuit testing to the field of circuit design. When “imperfect” algorithms and circuit structures are employed, a substantial yield for an error-tolerant digital circuit, in terms of power consumption, speed performance, and transistor count, can be realized.

Adopting the ideas and techniques in “error-tolerant” technology in the design of digital adders, a novel and innovative type of adder—Probabilistic Adder for Error Tolerant Application (ETA) has been designed and this is the major contribution of this report. The incentive to design such a new type of adder using the emerging technologies lies in the fact that adder is the most critical arithmetic block in computational systems and is always the dominant factor in determining the overall performance of a system. For modern computational systems, the increasingly huge data set and the need for instant response require the adder to be large and fast. Meanwhile, as portable digital devices become more and more popular, the requirement on power consumption has also become rigorous. The conventional Ripple-Carry Adder consumes very low power, but its speed performance hinders its employability in high-speed systems. The Carry-Lookahead Adder has excellent speed performance due to its intrinsic advantage in eliminating the carry propagation. However, its characteristics of high power consumption and large circuit area render it not suitable for use in low power systems. As a matter of fact, one of the limitations in conventional digital circuit design is the trade-off between power consumption and speed performance. Obtaining higher speed usually means more power will be consumed and lower power will normally degrade the speed of a circuit. To breakthrough this bottleneck in conventional technologies for designing a truly low-power and high-speed digital circuit, a new metric besides **power** and **speed** should be brought into the design process. In the proposed designs, **accuracy**

plays the role of such a new metric. By sacrificing some degree of accuracy, great improvements in both power consumption and speed performance can be achieved.

1.3 Objective

The objective of this work is to introduce a new type of adder—Probabilistic Adder for Error Tolerant Application (ETA). The implementations of the proposed ETAs (ETAII, ETAIII and ETAIV) will also be performed through HSPICE simulation. The simulation results of the ETAs will be compared with conventional adders to demonstrate the advantages of the proposed ETAs for both accuracy and performance. Following their successful implementations, other applications such as the Digital Signal Processing (DSP) circuits, e.g. imaging, and audio, will also be implemented to verify their functionality. Finally, the error tolerant concept will be extended to other digital modules such as the multiplier to realize even lower power and ultra high speed systems for future applications.

1.4 Contribution of the thesis

In this work, different types of addition algorithm of probabilistic adder for error-tolerant applications (ETAs) has been designed. The accuracies of the different addition algorithms for the various proposed designs have been analyzed. The proposed circuit designs are simulated using HSPICE software to determine their delay, power consumption and the area consumed. The error-tolerant applications which are suitable for the probabilistic adders are also studied. With the proposed ETAs embedded into the DSP applications, the results of the simulation have proven the merits and feasibility of the proposed adders in the signal processing applications.

1.5 Organization of the Thesis

The organization of the thesis is as follows: Chapter 2 provides a literature review on the existing conventional adders and some truncated circuits. This helps to set the background of the thesis and the key reasons for researching on the probabilistic adders. In Chapter 3, the very first probabilistic adder designed for the error-tolerant applications (ETA) is introduced. The idea is then expanded, leading to a few other probabilistic adders being designed. The detailed algorithms and circuit topologies of all the probabilistic adders designed will be described in depth in the chapter. Further, the accuracy, power, speed comparisons are specified to show the merit and shortcomings of each of the proposed probabilistic adders. In Chapter 4, the applications of these probabilistic adders are verified. Both image and sound applications had been studied to attest its “error-tolerant” viability. The results of the application proved that the probabilistic adders proposed are not only acceptable but provided enhanced speed performance and huge savings in power. Finally, in Chapter 5, a conclusion is drawn for this work and the direction for future research is also presented.

Chapter 2: Literature review

The concept of Error Tolerant in digital VLSI (very-large-scale integration) design will first be reviewed before the introducing the proposed adders. To aid the discussion, the existing conventional adders have been summarized in the chapter and a brief review of some existing probability logic systems will also be discussed.

2.1 Error-Tolerance

2.1.1 Concepts

In conventional digital VLSI design, a usable circuit or system with no defects is usually assumed to be perfect and can always provides precise and accurate results. But such perfection is seldom found in the real world. The real world always accepts “analog computation”, which generates “good enough” results rather than totally accurate results [9]. In fact, for many digital systems, the data they process may already contained errors. For example, in a communication system, the analog signal coming from outside world is first sampled and quantized to digital data on the front end, then the digital data are processed and transmitted through a noisy channel, and finally, the digital data are converted back to analog signal at the back end. During this process, errors may occur along the way. Since it is impossible or difficult to constantly maintain the correct data or results, it may be better for users to be more “generous” to accept certain amount of errors. This is the basic idea of Error-Tolerance.

According to the definition given in [10], a circuit is error-tolerant with respect to a specific application, if (1) it contains defects that cause internal and may cause external errors, and (2) the system that incorporates this circuit produces acceptable results. When incorporates the error-tolerant circuit, a digital system is no longer totally “correct”. Instead, certain errors may be generated in the output. This “imperfect” attribute seems to be not too appealing. However, the need for the error-tolerant circuit was foretold in the 2003 International Technology Roadmap for

Semiconductors (ITRS) [7]. It was quoted that: “Relaxing the requirement of 100% correctness in both transient and permanent failures of signals, logic values, devices, or interconnects may reduce the cost of manufacturing, verification and testing.”

2.1.2 Integrated Circuit Testing Methodology that Support Error-Tolerance

The original concept of Error-Tolerance is derived from the perspective of circuit testing, so several testing methodologies that support error-tolerance have been proposed and developed [11, 12, and 13]. Although the testing methodology is not the concern of our work, the ideas, attributes, and analysis methods proposed in these work help us build a better view of error-tolerant digital integrated circuits design, which is the main contribution of this thesis.

In conventional integrated circuit testing techniques, the targets of testing are all possible faults that may occur in the circuit. However, in the error-tolerance supported testing methodology, the targets of testing are reduced to only the **unacceptable** faults that are predetermined by designer/user.

An important attribute that has been proposed in the error-tolerance supported testing is the error-rate. It is defined as the fraction of incorrect results that a system produces [14]. Figure 2.1 shows an error-rate based testing methodology that supports error-tolerance [12]. In this methodology, each individual fault in the target circuit has a corresponding error-rate that quantitatively indicates the probability that the specific fault happens in the target circuit. For every error-tolerance supported system, there is a maximum acceptable system error-rate specified by the designer/user. Those faults whose error-rates are higher than the maximum acceptable system error-rate are considered as unacceptable faults while the rest faults are expected to be tolerated by the system. The idea and attribute described in the error-tolerance supported testing methodology are actually the prototype of the idea and attribute that will be employed in the probabilistic adder for error-tolerant applications (ETA) design.

Figure 2.1 Error-rate Based Testing Methodology

2.1.3 A Case Study of Error-Tolerance

A framework for the analysis of the applicability of the Error-Tolerance technique is presented in [15]. The framework is illustrated with respect to a digital telephone-answering device (DTAD).

The target system of DTAD has two main components: the microcontroller and the flash memory, which is assumed to be defective. In the proposed framework, the relationships between the *defect density (error-rate)*, the *acceptable performance*, and the *effective yield* are investigated. The *defect density* is defined as the ratio between the number of faults and the size of the flash memory. The *acceptable performance* is referred to the performance (subjective or objective) that is acceptable to the user according to certain measurement standard. The *effective yield* represents the yield in manufacturing process due to the employment of Error-Tolerance technique.

A brief introduction of the working mode of the DTAD is given as follow. In the answering mode, the ADC device in the system samples and quantizes the speech signal, the codec encodes this quantized signal, and the output bit-stream is stored

in the flash memory. When the user listens to the recorded speech, the microcontroller extracts the encoded data stored in the memory, and the codec decodes the data and finally recovers the speech.

Because the flash memory employed in the DTAD is defective, the quality of the output of this system is degraded. If the “imperfect” output is acceptable to the user according to certain measure standard, this system can be regarded as an error-tolerant system.

The fault model considered in [15] is the multiple stuck-at fault model. The erroneous bits in the memory are either stuck-at-1 or stuck-at-0. Faults are randomly allocated through the memory based on the uniform distribution. Then twenty different fault densities between 0% and 1% are simulated. For each fault density, fifty different random distributions of faults are considered.

To measure the quality of the performance of the target DTAD, a kind of subjective test whose guidelines form a *mean opinion score* [16] is conducted to the simulation results. The qualitative interpretations of the *mean opinion score* are: 1 (bad), 2 (poor), 3 (fair), 4 (good), 5 (excellent). According to [15], if the acceptance threshold value T , which is the lowest acceptable *mean opinion score*, is set to 3 (fair), the corresponding acceptable fault density for the DTAD is 0.20%. That means when 0.20% of all the bits in the flash memory are defective, the whole system still has acceptable performance. The resulting yields for this error-tolerant DTAD can reach to around 75%, which is a substantial improvement.

2.2 Conventional Designs of Digital Adders

2.2.1 Half adder and Full adder

A half adder accepts two input bits (A and B) and generates two output bits, sum (S) and carry-out (C_o). Table 2.1 is the truth table for a half adder. The Boolean expressions are given in Equations (2.1) and (2.2):

$$S = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B} \quad (2.1)$$

$$C_o = A \cdot B \quad (2.2)$$

The logic structure of a half adder is shown in Figure 2.2.

Table 2.1 Truth table for half adder

A	B	S	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 2.2 Logic structure of half adder

Table 2.2 Truth table for full adder

A	B	C _i	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A full adder takes 3 inputs, two addend bits (A and B) and a carry-in bit (C_i), and, like the half adder, generates 2 outputs, sum (S) and carry-out (C_o). The truth table for a full adder is given in Table 2.2.

According to the truth table of Table 2.2, the Boolean expressions for the full adder can be derived as follows:

$$\begin{aligned}
 S &= A \oplus B \oplus C \\
 &= \overline{A} \cdot \overline{B} \cdot C_i + \overline{A} \cdot B \cdot \overline{C_i} + A \cdot \overline{B} \cdot \overline{C_i} + A \cdot B \cdot C_i
 \end{aligned}
 \quad (2.3)$$

$$C_o = A \cdot B + A \cdot C_i + B \cdot C_i \quad (2.4)$$

For many implementation strategies, such as the Carry-Lookahead Adder, the intermediate signals, G (generate), D (delete), and P (propagate) are needed in the design processes. These three intermediate signals are defined as follows:

$$G = A \cdot B \quad (2.5)$$

$$D = \overline{A} \cdot \overline{B} \quad (2.6)$$

$$P = A \oplus B \quad (2.7)$$

With the above, the expressions for S and C_o can be written in terms of P and G :

$$S = P \oplus C_i \quad (2.8)$$

$$C_o = G + P \cdot C_i \quad (2.9)$$

One possible logic structure of a full adder is shown in Figure 2.3. A variety of implementations of the full adder with different circuit structure, transistor count, and performance can be found. Figure 2.3 provides two different implementations of a full adder. Figure 2.4 (a) is the conventional 28-transistor full adder (28T) which is a complementary CMOS circuit derived directly from the logic equation [17]. The drawbacks of the 28T adder are that it consumes a large circuit area and its speed is slow. Figures 2.4 (b) shows the Transmission Gate Adder (TGA) [18]. All these circuits can be implemented using minimum-sized transistors.

Figure 2.3 Logic structure of full adder

(a) 28-transistor full adder

(b) Transmission gate full adder

Figure 2.4 Different implementation of full adder

2.2.2 Ripple-Carry Adder

Ripple-Carry Adder (RCA) [17] is the simplest architecture of an adder. An N -bit RCA is constructed by simply cascading N full adders in series. The carry-out signal of one full adder serves as the carry-in signal of the next full adder, i.e., $C_{o,k} = C_{i,k+1}$, where $0 \leq k \leq N-2$. The structure diagram of the RCA is given in Figure 2.5.

Because of its simple and regular structure, RCA can consume less power and occupies smaller area than any other conventional adders. However, the time delay of this architecture can be enormous. In the worst case, the carry signal will be propagated from the Least Significant Bit (LSB) all the way to the Most Significant Bit (MSB). So the critical path in RCA is the entire carry propagation chain. The delay time is linearly proportional to the total number of full adders, N . Thus, RCA is regarded as the slowest adder among all conventional adders and it cannot meet the rigorous requirement on circuit/system speed in today's technology.

Figure 2.5 Ripple-Carry Adder

To shorten the critical path of the adder, many techniques have been developed. In the following subsections, several improved architectures of the adder are presented.

2.2.3 Carry-Skip Adder

Carry-Skip Adder (CSK) [19] is also named as Carry-Bypass Adder. Its concept can be illustrated by Figure 2.6. For a 4-bit adder module, an additional connection between the carry-in signal $C_{i,0}$ and the carry-out signal $C_{o,3}$ is added to the normal carry propagation path via a multiplexer. When all the propagation signals P_k ($k=0, 1, 2, 3$)

in such a module are high (i.e., $P_0P_1P_2P_3=1$), the carry-in signal $C_{i,0}$ is forwarded immediately to the next block as the carry-out signal $C_{o,3}$, by skipping the whole propagation path in this block. If this is not the case, the carry-out signal is obtained through the normal carry propagation path. The block diagram of a 16-bit CSK is given in Figure 2.7. The critical path of the adder is shaded in gray in the figure.

Figure 2.6 4-bit Carry-Skip Adder

Figure 2.7 16-bit Carry-Skip Adder

2.2.4 Carry-Select Adder

The major problem of Ripple-Carry Adder (RCA) is that each full adder cell has to wait for the carry signal to be fed from the previous stage before a correct carry-out

signal can be generated. The idea of Carry-Select Adder (CSL) [20] is to consider the two possible values of the carry-in signal, to generate the carry-out signals for both possibilities in advance. Once the “real” value of carry-in is known, the correct result will be selected through a simple multiplexer stage. Figure 2.8 demonstrates an implementation of the CSL. It can be seen that the whole adder has been divided into a number of equal-length adder stages. For each stage, instead of waiting for the arrival of the carry generated by the previous stage, both the “0” and “1” possibilities are evaluated. When the carry-in signal finally settles, either of the two possible results is selected and passed to the next stage. In this way, the critical path is greatly shortened compared with the RCA.

Figure 2.8 Linear Carry-Select Adder

The structure shown in Figure 2.8 can actually be further optimized. For each multiplexer, there are three inputs, two pre-calculated carry signals that serve as the candidates to be selected and the real carry signal coming from previous stage that plays the role as a control signal. It can be observed that there exists a mismatch between the arrival times of those signals. The outputs of the two parallel carry-generation blocks are stable long before the control signal arrives. To equalize these two propagation paths, the full adder stages can be built in a progressive-sized manner instead of the equal-sized manner. The modified structure is illustrated in Figure 2.9. In the original structure, each stage contains the same number of full adder cells. The delay time of this structure is linearly proportional to the size of the adder, N , so the adder with this structure is called Linear Carry-Select Adder (LCSL) [17]. On the other hand, in the modified structure shown in Figure 2.9, each stage

contains different number of full adder cells and the number increases by one from one stage to the next. The delay time of the modified structure is proportional to $\sqrt{N}$ instead of N , so the adder with the modified structure is called Square-Root Carry-Select Adder (SRCSL) [17].

Figure 2.9 Square-Root Carry-Select Adder

The major problem of the CSL is that an additional set of carry generation circuits is needed so that the whole circuit consumes more power and occupies more area.

2.2.5 Carry-Lookahead Adder

In the CSK and CSL described above, the carry-rippling effect still exists even though they have shortened the critical path in one way or another. To design even faster adders, this carry-rippling effect should be totally eliminated. According to Equations (2.8) and (2.9), the following relation holds for the k -th bit position in an N -bit adder.

$$C_{o,k} = G_k + P_k C_{i,k} = G_k + P_k C_{o,k-1} \quad (2.10)$$

By recursively applying Equation (2.9), the following fully expanded form can be obtained:

$$C_{o,k} = G_k + P_k (G_{k-1} + P_{k-1} (\dots + P_1 (G_0 + P_0 C_{i,0}))) \quad (2.11)$$

The sum on the k -th bit position can then be expressed as follow:

$$\begin{aligned}
S_k &= P_k \oplus C_{o,k-1} \\
&= P_k \oplus (G_{k-1} + P_{k-1}(G_{k-2} + P_{k-2}(\dots + P_1(G_0 + P_0 C_{i,0})))) \quad (2.12)
\end{aligned}$$

From Equations (2.11) and (2.12), it can be seen that the carry-out bit and sum bit on any bit position can be derived with just the input bits, without involving any internal carry signals. Thus, theoretically speaking, all the sum bits can be generated simultaneously, and almost immediately after receiving the inputs. In this way, the carry propagation path is totally eliminated. The adder derived from Equations (2.11) and (2.12) is named Carry-Lookahead Adder (CLA) [21]. The block diagram of a 4-bit CLA is depicted in Figure 2.9. One of many possible implementations of a 4-bit CLA is shown in Figure 2.12 [18].

While the CLA is superior in speed performance, its costs in power consumption and circuit area are tremendous. When the size of the adder, N , increases, the power consumption and circuit area of the adder will increase dramatically. So, the carry-lookahead structure shown in Figure 2.10 is only suitable for small adders (usually, $N \leq 4$).

To construct large adders, several techniques have been proposed. The simplest way is to use the carry-lookahead technique to construct a number of 4-bit adders and then cascading these 4-bit adders in the ripple-carry way to form the large adder (illustrated in Figure 2.12). Because this design strategy contains two techniques, carry-lookahead technique and ripple-carry technique, it can also be called *hybrid adder* (Note that the term *hybrid adder* can be referred to any design scheme that makes use of two or more design techniques.). This hybrid adder combines the characteristics of both CLA and RCA, so it achieves a balance between high speed performance and low power consumption.

Figure 2.10 Block diagram of 4-bit Carry-Lookahead Adder

Figure 2.11 Implementation of 4-bit Carry-Lookahead Adder

Figure 2.12 N-bit Carry-Lookahead Adder constructed in the ripple-carry way

2.3 Existing Probabilistic/truncated system

2.3.1 Carry State Estimation for Adders

In the work reported by Wallace *et al.* it was defined that the relationship between the probabilistic of the carry with other adder components [22]. It is based on the concept of predicting the carry from the least to the most significant halves of a 32 or 64 bit adder in such a way it has a high probability of being correct, while introducing only a low area overhead from the required early completion control circuitry.

In this article [22], a 32-bit adder is separated into two parts: most significant part (MS) and least significant part (LS), as shown in Figure 2.13. The MS adder is duplicated to cover the two possibilities for the carry, from the LS adder of a one or zero, and the result is multiplexed depending on the true state of the LS adder carry output. The circuit will be faster since the MS part and LS part can be processed concurrently; however, the disadvantage is the significant additional circuit complexity, and hence the area and power consumed.

Figure 2.13 Carry Select Adder

In Wallace's design, the "estimated carry structure" (ESTC) uses a carry select adder (CSA) structure but one of the most significant adders and multiplexer are dispersed with completely. The circuit only uses the most significant (MS) bits of the least significant (LS) half of the operands to predict the carry from the LS to MS halves of the adder. Since the probability of a correct prediction turns out to be 0.75, parallel addition can be used for the LS and MS halves of the data for 75% of the additions, thereby reducing carry ripple time.

By designing the statistical probability of a carry to be in a particular state, a 32 bit adder can be constructed which, for the majority of additions, will operate as a 16 bit parallel adder. The design has a good delay-area product than the conventional adders.

This proposed design adopts the concept of having probability in the carry signal. Although the adder is not for an error-tolerant application (since the carry bit will finally pass through a selector and hence the value is fixed), it can do some trade-off between the probability, delay and area.

2.3.2 Probabilistic Multiple Carry Estimates

It has been demonstrated by [23] that using the statistical probability of a carry when it is in a particular state, a 32-bit adder can be constructed in which for the majority of additions there is an improvement in the speed performance of the adder.

Basically, the work in [23] is similar to that reported in [22]. The major improvement is that it derives the relationship between the probability of a correct prediction and the carry, to derive its enhanced output as illustrated in the better probability of Figure 2.14.

Figure 2.14 Probability vs. Carry for prediction

According to Figure 2.14, there is a significant increase in the probability of having a correct prediction when four to five carries is engaged in the prediction. Thereafter, the increase is only minimal.

With this concept, the design uses multiple carries to predict a correct carry from the LS to the MS half of a 32-bit adder. As each new carry is introduced for prediction,

the probability will be increased. This is at the expense of other components such as the timing and area.

Although this design is still not sufficient for an error-tolerant application, the concept of having a relation between probability and carry for predicting is favored and is therefore adopted in our design.

2.3.3 Probabilistic Ripple-Carry Adder

A mathematical model on probabilistic ripple-carry adder has been proposed by [24]. The model gives explicit expressions for calculating the error probabilities of sum and carry bits.

Presented in Figure 2.15 is an error probability versus the voltage supply of their reported circuit [24].

Figure 2.15 Error probabilities obtained by theoretical value and by simulations

It is well known that the lower the supply voltage, the lower the power consumption. One way to design the low-power circuit is thus to reduce the supply voltage. However, with the voltage decreasing, the noise will become significant. When the supply voltage decreases to below the threshold value, the error will be introduced.

This design presents the model representing a relationship between the error rate and supply voltage, and how the errors are propagated. It proves that the error-rate (accuracy) and the voltage (power) can be the mutual trade-off components. In the author's proposed work, the accuracy, the power and the delay will be traded to construct probabilistic adders for error-tolerant applications.

2.4 Summary

In this Chapter, the definition and the concept of Error-Tolerant is described. Other than the circuit testing work reported, the concept had also been extended to circuit design whereby a mathematical model on probabilistic ripple-carry adder has been proposed by [24], allowing explicit expressions for calculating the error probabilities of sum and carry bits. The conventional adders are also reviewed in the chapter, starting with the half adder and full adder, followed by the Ripple Carry Adder (RCA), Carry Skip Adder (CSK), Carry Select Adder (CSL), and lastly, the Carry Lookahead Adder (CLA). Their algorithms and architectures are described and their advantages/disadvantages are elaborated. Consequently, the currently existing probabilistic systems are also discussed.

Chapter 3: Probabilistic Adders

3.1 Define of terms

Before discussing on the design of the ETAs, the exact definitions and explanations of some commonly used terminologies in this thesis are given as follows:

- *Overall error (OE)*. It is defined as the difference between the correct result and the obtained result. It can be computed by using the following equation: $OE = |R_c - R_e|$, where R_e is the result obtained by the adder, and R_c denotes the correct result (both results are represented as decimal numbers).
- *Accuracy (ACC) of adder*. In the scenario of error-tolerant design, the accuracy of an adder is used to indicate how “correct” the output of an adder is. It is defined as $ACC = (1 - \frac{OE}{R_c}) \times 100\%$. Its value ranges from 0% to 100%. According to the mathematical expression, it can be seen that the accuracy of an adder is depending on the output result so that is not a constant. Actually, the accuracy of an adder can be regarded as a variable with respect to the output/input pattern and its value is equal to the accuracy of a specific obtained output. In this thesis, for convenience, the term “accuracy” is sometimes used to denote both the accuracy of an adder and the accuracy of its output.
- *Minimum acceptable accuracy (MAA)*. Although some errors are allowed to exist in the output of an ETA, the accuracy of an acceptable output should be “high enough” (higher than a threshold value) to meet the requirement of the whole system. Minimum acceptable accuracy is just that threshold value. The obtained results whose accuracy is higher than the minimum acceptable accuracy are called acceptable results. The value of the minimum acceptable accuracy is often preset by the customers/designers according to specific applications.

- **Acceptance probability (AP).** Since the accuracy of an adder is dependent on the output/input pattern and the outputs/inputs of a digital system are often regarded as random signals, the accuracy of an adder can also be taken as a random variable. Acceptance probability is the probability that the accuracy of an adder is higher than the minimum acceptable accuracy. It can be expressed as $AP = P(ACC > MAA)$, where value P = probability and AP ranges from 0 to 1. This parameter is usually used as the key metric indicator in determining the accuracy performance of an ETA.

3.2 Previous work: ETA Type I

3.2.1 Addition Algorithm

In a conventional adder circuit, the delay is mainly attributed to the carry propagation chain along the critical path, from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). Moreover, a significant proportion of the power consumption of an adder is due to the glitches that are also caused by the carry propagation. Therefore, if the carry propagation can be eliminated or curtailed, a great improvement in both the speed performance and power consumption can be achieved. This algorithm can be illustrated via an example given in Figure 3.1.

Figure 3.1 Addition algorithm for ETAI

First the input operands are split into two parts: an accurate part that includes a number of higher order bits and an inaccurate part that is made up of the remaining lower order bits. The lengths of each part need not necessarily be equal. The addition process starts from the middle (joining point of the two parts) towards the two opposite directions simultaneously. In the example, the two 16-bit input operands, $A = "1011001110011010"$ (45978) and $B = "0110100100010011"$ (26899), are divided into two equal-sized parts, each of which contains 8 bits.

For the higher order bits of the input operands that fall into the accurate part, the operation is performed from right to left (LSB to MSB) and normal addition method is applied. This segment is named the *accurate part* because it follows the conventional accurate addition algorithm. For the example shown in Figure 3.2, the partial sum generated in the accurate part is "100011100", which is perfectly correct.

For the lower order bits of the input operands that fall into the inaccurate part, a special addition mechanism is applied. In this part, no carry signal will be generated or taken in at any bit position such that the carry propagation path no longer exists. To minimize the overall error caused by eliminating the carries, a special strategy is adopted. Its operational process is described as follow: check every bit position from left to right (MSB to LSB); and on a bit position, if either of the two input operand bits is "0", normal one-bit addition is performed to derive the sum bit on that position and the operation proceeds to next bit position; if both of the input bits are "1", the checking process is stopped and from this bit onwards, all the sum bits are set to "1". In this way, the overall error generated due to the elimination of carry bits can be reduced to minimal. In the example, at the fifth bit position, the two input bits, A_4 and B_4 , are both equal to "1", so all the sum bits on its right are set to "1". The partial sum generated in the inaccurate part is therefore "10011111", which contains error.

The final result of the complete addition is therefore "1000111001001111" (72863). This is the result obtained using the proposed addition algorithm. On the other hand, the correct result of this addition, which can be derived using the normal addition

algorithm, is “10001110010101101” (72877). So the overall error generated in this example is:

$$OE = 10001110010101101 (72877) - 10001110010011111 (72863) = 1110 (14).$$

The accuracy of the adder with respect to these two input operands is:

$$ACC = (1 - \frac{14}{72877}) \times 100\% = 99.98\%.$$

In this new addition method, the carry propagation only exists in the accurate part. The accurate part is constructed in the conventional way because the higher order bits of a result need to be made as accurate as possible, as they play a more important role (have higher weights) than the lower order bits do. By eliminating the carry propagation path in the inaccurate part and performing the addition in two separate parts simultaneously, the overall delay time is greatly reduced and so is the power consumption.

3.2.2 Hardware Implementation

Block diagram of the hardware implementation of ETAI is provided in Figure 3.2. This most straightforward structure consists of two parts: an accurate part and an inaccurate part. The accurate part, which contains $n-m$ bits, is constructed using a conventional adder such as the RCA, CSK, CSL or CLA. The carry-in of this adder is connected to ground. The accurate part is used to compute the higher order bits of the sum. The inaccurate part, whose size is m -bit, constitutes two blocks: a carry-free addition block and a control block. The carry-free addition block generates the sum bits on the lower order bit positions. The control block is used to generate the control signals to determine the working mode of the carry-free addition block. (For the detail information, please refer to reference [30].)

Figure 3.2 Block diagram of the hardware implementation of ETA Type I

3.2.3 Simulation Results

The ETAI is simulated along with four different types of conventional adders including the RCA, CSK, CSL, and CLA, using HSpice. All the circuits are implemented using Chartered Semiconductor Manufacturing Ltd's 0.18- μm CMOS process. The input data rate is set at 100 MHz. The HSpice schematic simulation results are all tabulated in Table 3.1.

Table 3.1 Simulation Results

Type	Power (mW)	Delay (ns)	PDP (pJ)	PDP Saving (%)	Transistor Count
RCA	0.22	4.04	0.89	66	896
CSK	0.46	2.90	1.33	77	1728
CSL	0.60	3.06	1.84	84	2176
CLA	0.51	2.37	1.21	75	2208
ETAI	0.13	2.29	0.30	-	1006

For extracting the results, we use the HSPICE software to build the model of our proposed ETAI. Then we randomly generated 100 sets of inputs using C program `random( )` function. For each set of input, we ran the simulation for each type of the

adder and recorded the power consumption. With the 100 sets of power consumption results, an average power consumption of the ETAI was determined. For the delay of the ETAI, the author calculated the worst case input and used it to simulate the delay. For the transistor count of the ETAI, it can be directly derived from the HSPICE software. As for the other conventional adders, the result extracting steps are similar to that of the ETAI. Figure 3.3 shows the flow chart of the result extracting of the ETAI.

Figure 3.3 Flow chart of result extraction of ETAI

Comparing the simulation results of the proposed ETAI with the conventional adders, it is evident that ETAI performed the best in terms of power consumption, delay, and Power-Delay Product (PDP). The PDP of the ETA is noted to be 66%, 77%, 84%, and 75% better than the RCA, CSK, CSL, and CLA, respectively. Besides, when comes to transistor count comparison, the proposed ETAI is almost as good as the RCA.

3.2.4 Problems of ETAI

ETAI has a problem of calculating small number inputs. Take for example $0000000000001111 + 0000000000001111$, ETAI will yield a result of 0000000000001111 which is far from the correct value of 00000000000011110 . Although most time input operands are randomly distributed over all the possible numbers and this distribution follows the uniform distribution (It means every input

pattern occurs with the same probability. For example, for a 32-bit adder, its input operand can be any number from 0 to $2^{32}-1$, and each number has the same possibility to occur), there are some applications, the inputs are not evenly distributed, sometimes the small numbers play the major role.

For a 32-bit ETAl, the values of AP associated with different MAA's and input ranges are tabulated in Table 3.2. The columns represent the AP values associated with the input operands in four different ranges, and the rows show the AP values based on different MAA's from 90% to 100%.

Table 3.2 AP of ETAl with different MAA's and input ranges

Input range	$0 \sim 2^8 - 1$	$0 \sim 2^{16} - 1$	$0 \sim 2^{24} - 1$	$0 \sim 2^{32} - 1$
MAA=100%	0.0988	0.0096	0.0032	0.0025
MAA=99%	0.1364	0.1491	0.5650	1.0
MAA=98%	0.1897	0.2055	0.7324	1.0
MAA=97%	0.2239	0.2472	0.8375	1.0
MAA=96%	0.2656	0.2808	0.9014	1.0
MAA=95%	0.3002	0.3124	0.9385	1.0
MAA=94%	0.3344	0.3409	0.9570	1.0
MAA=93%	0.3568	0.3614	0.9679	1.0
MAA=92%	0.3858	0.3899	0.9754	1.0
MAA=91%	0.4040	0.4140	0.9812	1.0
MAA=90%	0.4266	0.4318	0.9856	1.0

From the table, it is clear that, for small input operands, the accuracy performance of the proposed ETAl is very poor. As the inputs become larger, the AP values also become larger. When the input operands are large enough, the AP values of the proposed ETAl are almost equal to "1". The simulation results prove that the proposed ETAl is only suitable for large input operands.

3.3 ETA Type II

Different from ETAl, the Probabilistic Adder for Error Tolerant Application Type II (or ETAlI for short), does not eliminate the entire or part of the carry propagation path. Instead, it splits the entire carry propagation path into a number of short paths and completes the carry propagations in these short paths concurrently. In this way, the speed performance of the adder is drastically improved and almost without degrading its power consumption. However, just like ETAl, the accuracy of the adder may be affected due to the performance improvement.

3.3.1 Theoretical Analysis

Consider the binary number addition process, the carry signal at each bit position is determined by all the previous input bits. For the worst case, this carry signal is generated at the LSB and propagates along the carry chain to the current bit position. If this worst case happens, a lot of time and power will be consumed along the carry propagation path. But in fact, the worst case seldom happens. For most of the cases, this carry signal can be determined by just several input bits on the right of the current bit position. Assume that the input operands are perfectly random; the probabilities of getting correct carry signal on the i -th bit position when different numbers of bit positions are involved in determining this carry signal can be derived. Some of the results are tabulated in Table 3.3. As indicated in the table, if the number of input bits to the CLA is just 1 bit, the correct carry signal will be correct at 50% of the time; if the number of bits to be considered is increased from 1 to 6 (see Table I), the probability of having a wrong result is less than 2%; and if more than 8 bits are taken in to consideration, the probability of getting a wrong signal will be less than 0.5%. A simple formula to represent such outcome is $P = 1 - 0.5^n$, where n is number of neighboring bits considered.

**Table 3.3 Probability of getting correct carry signal when different
number of bit positions are taken into consideration**

Number of bit positions considered	1 bit	2 bits	4 bits	8 bits	12 bits
Probability of getting correct carry signal	≥ 0.5	≥ 0.75	≥ 0.9375	≥ 0.9844	≥ 0.9961

Of course, everyone hopes to get a perfectly correct result in any case. The problem is how much (in terms of power and time) one has to sacrifice to remove the incorrect results that may occur only with very small probabilities and whether it is worthy to do so. Just as discussed in previous chapters, if some errors are allowed to exist; great achievements in power- and time-saving can be obtained.

3.3.2 Architecture of ETAIL

The architecture of ETAIL is depicted in Figure 3.4. For an N-bit adder, it is divided into M ($M \geq 2$) blocks, each of which contains N/M bits. In each block, there are two separate circuitries. One is to generate the carry-out signal based **ONLY** on the input bits in this block (i.e., it does not take in the carry signal from previous block), and the other takes the carry-in signal from previous block and generates all the sum bits in this block. In this manner, the carry propagation only exists between two neighboring blocks instead of lying along the entire adder structure. In other words, the longest carry propagation path of ETAIL is $2N/M$ -bit long. The worst-case delay path is shaded in gray in Figure 3.4. So, the worst-case delay of the ETAIL is only $2/M$ times of the conventional adder. Again, as the carry propagation path has been curtailed, the dynamic power consumption of the adder caused by the spurious switching activities is also reduced. However, this power reduction can be offset by the additional carry generators [31].

Figure 3.4 Architecture of ETAIL

3.3.3 Dividing Strategy

Similar with the design of ETAl, the dividing strategy may affect the overall performance of the adder. In ETAIL design, the dividing strategy refers to the choice of the number of blocks the adder is divided into. If fewer blocks are engaged and thus more bits are contained in one block, the probability of getting correct results becomes higher while the delay path also becomes longer; on the contrary, if the adder is divided in a “finer” module (i.e., more blocks are engaged), the speed performance can be increased while the possibility that incorrect results occur becomes higher. Take a 32-bit ETAIL for instance, the simulation results of the AP under different dividing strategies and MAA’s are provided in Table 3.4. These simulations are completed by a C program. 10000 random input patterns in the range of $0 \sim 2^{32} - 1$ are tested. For the C program calculating the AP, please refer to Appendices A – C for the respective Codes for calculating the AP for ETAIL.

Table 3.4 AP of ETAIL under different dividing strategies and MAA's

	M = 1 bit	M = 2 bits	M = 4 bits	M = 8 bits	M = 16 bits
MAA=100%	0.0118	0.2204	0.8306	0.9961	1.0
MAA=99%	0.5238	0.7927	0.9725	0.9985	1.0
MAA=98%	0.5419	0.8075	0.9795	0.9985	1.0
MAA=97%	0.5534	0.8119	0.9815	0.9995	1.0
MAA=96%	0.5706	0.8236	0.9819	0.9995	1.0
MAA=95%	0.5864	0.8352	0.9833	1.0	1.0
MAA=94%	0.5894	0.8367	0.9833	1.0	1.0
MAA=93%	0.5973	0.8386	0.9838	1.0	1.0
MAA=92%	0.6062	0.8401	0.9838	1.0	1.0
MAA=91%	0.6155	0.8425	0.9843	1.0	1.0
MAA=90%	0.6213	0.8468	0.9843	1.0	1.0

3.3.4 Implementation of a 32-bit ETAIL

The block diagram of the 32-bit ETAIL implementation is depicted in Figure 3.5. Note that when compared with Figure 3.4, the carry generator in the highest (leftmost) block has been eliminated, because this circuit is actually a dummy in this structure. In the proposed design, the whole adder is divided into 8 blocks, each of which has 4 bits. The carry generator is implemented employing the carry-lookahead methodology. The sum generators are realized as the most simple ripple-carry adders for the purpose of power saving.

The circuit simulation was performed using HSPICE, and the results are tabulated in Table 3.5. All the simulation parameters are the same with those described in Table 3.1.

Figure 3.5 Implementation of a 32-bit ETAIL

Table 3.5 Simulation results of ETAIL

Type	Power (W)	Delay (S)	PDP (J)	TC
ETAIL	2.4E-04	9.5E-10	2.3E-13	1372

3.3.5 Accuracy of ETAIL for small number inputs

As concluded in Section 3.2.4, when the inputs are of small numbers, the accuracy performance of the proposed ETAIL will be very poor. To test if this situation exists in ETAIL, the same simulations were performed on ETAIL and the results are provided in Table 3.6.

From Table 3.6, it is obvious that the accuracy performance of the adder for small input operands has been significantly improved using the new method, although the accuracy performance for large input operands is now degraded (compare the last column of Table 3.2 and Table 3.6). It can be seen that there is no big difference of the AP values between different input ranges using the new method. In other words, the unacceptable results are distributed over the whole input range from 0 to $2^{32} - 1$ instead of “squeezing” in the range of small numbers.

Table 3.6 AP of ETail with different MAA's and input ranges

Input range	$0 \sim 2^8 - 1$	$0 \sim 2^{16} - 1$	$0 \sim 2^{24} - 1$	$0 \sim 2^{32} - 1$
MAA=100%	0.9691	0.9090	0.8590	0.8389
MAA=99%	0.9691	0.9408	0.9393	0.9699
MAA=98%	0.9691	0.9414	0.9417	0.9780
MAA=97%	0.9691	0.9414	0.9422	0.9824
MAA=96%	0.9691	0.9439	0.9453	0.9829
MAA=95%	0.9691	0.9493	0.9517	0.9829
MAA=94%	0.9691	0.9545	0.9557	0.9829
MAA=93%	0.9691	0.9582	0.9598	0.9829
MAA=92%	0.9691	0.9612	0.9624	0.9834
MAA=91%	0.9691	0.9617	0.9636	0.9834
MAA=90%	0.9691	0.9638	0.9663	0.9839

3.3.6 Modified ETail

The structure of ETail introduced in Section 3.3.4 is designed in an unbiased manner, which has a good performance in both power and speed. However, the degraded accuracy performance of this structure for large input operands that has been indicated in Section 3.2.5 may restrict its usage in some applications. A biased structure can be implemented to further improve the accuracy performance of ETail: the high order bits should be more accurate than the low order bits as they play a more important role in representing a number. Therefore, for the several higher order bit positions in an adder, more input bits should be taken into consideration when calculating the carry signals. Figure 3.6 depicts the modified structure of the 32-bit ETail (for convenience, the modified ETail is named ETaiIM). In this structure, the first three carry generators are cascaded to generate the carry signals for the highest 2 blocks. In this way, the carry signal for the highest block is generated by the following 12 bits and the one for the second block is generated by 8 bits. The rest of the circuit is still the same with that shown in Figure 3.5. The accuracy performance

of this modified structure is given in Table 3.7. The circuit implementation of ETAIIM has been simulated and the results are given in Table 3.8. Compared with ETAIL, the delay time has increased significantly. This degradation of speed performance can be regarded as a trade-off due to improvement in accuracy.

Figure 3.6 Structure of 32-bit ETAIIM

Table 3.7 Accuracy performance of the ETAIIM

MAA	AP	MAA	AP
100%	0.8854	95%	1.0
99%	0.9990	94%	1.0
98%	0.9995	93%	1.0
97%	1.0	92%	1.0
96%	1.0	91%	1.0

Table 3.8 Simulation results of ETAIIM

Type	Power (W)	Delay (S)	PDP (J)	TC
ETAIIM	2.4E-04	1.5E-09	3.6E-13	1372

3.3.7 Comparison with conventional adders

The comparison of circuit performance (i.e., the power consumption, delay time, PDP, and transistor count) between the proposed ETAIIM and conventional adders are illustrated in Table 3.9. From the table, it is clear that the proposed ETAIIM has

the best speed performance when compared to conventional adders. ETAIIM is also able to maintain low power consumption (almost as low as the RCA). Its PDP is 60%, 75%, 81%, and 71% better than that of the RCA, CSK, CSL, and CLA, respectively. Although the transistor count of the ETAIIM is larger than the RCA, it is better than other conventional adders. For the HSPICE code of ETAIIM for simulation, please refer to Appendix B – HSPICE Code of ETAIIM.

Table 3.9 Comparison between ETAIIM and conventional adders

Type	Power (W)	Delay (S)	PDP (J)	PDP Saving	TC
RCA	2.21E-04	4.04E-09	8.9E-13	60%	896
CSK	4.73E-04	3.02E-09	14.3E-13	75%	1760
CSL	6.09E-04	3.07E-09	18.7E-13	81%	2176
CLA	5.13E-04	2.37E-09	12.2E-13	71%	2208
ETAIIM	2.43E-04	1.47E-09	3.6E-13	--	1372

Comparing the simulation results of the proposed ETAIL and ETAIIM with those of the conventional adders and ETAI, it is evident that both ETAIL and ETAIIM can achieve a very small PDP that is almost the same as that of ETAI, which is of course far better than those realized using conventional adders. The transistor counts (areas) of ETAIL and ETAIIM are a little larger than the ETAI; but they are both able to solve the small input problem in ETAI.

And another point to be note is that ETAI has a much smaller probability of getting correct result, while ETAIL shows a much larger probability of correctness (as can be seen by comparing the second row of Tables 3.2 and 3.6).

3.4 ETA Type III

ETA Type I is good in power, speed and area. The only problem is its inability to evaluate small input numbers. To overcome this problem, an enhanced Probabilistic Adder for Error Tolerant Application – ETA Type III based on ETAI design is developed. ETAIII is able to maintain almost the same power and speed as that of ETAI and also able to calculate small input numbers that are much better than ETAI.

3.4.1 Architecture of ETAlII

ETAlII can be seen as a combination of ETAl and ETAlI. Similar to ETAl, ETAlII makes use of the combination of the dividing and conquering strategy. ETAlII is suitable for all input ranges which is contrary to ETAl, that is good only for large number addition. Similar to ETAlI, ETAlII divides the input bits into several blocks. The main idea of ETAlII is to divide the input bits into two parts – accurate part and inaccurate part, respectively. ETAlII is different from ETAl, in that the dividing is not fixed, but is decided by a circuit called “selector”. Then for the accurate part, the process uses the same as normal addition algorithm while for the inaccurate part, the process uses the method same as the inaccurate part in ETAl to get rid of the carry propagation. The architecture of ETAlII is shown in the Figure 3.7.

on the range of the input. In order to solve this problem, ETAll takes a special dividing method that is determined by the range of the input number, and hence permitting flexibility. To be specific, the range of input numbers will first be examined, followed by the selection of the first three non-zero blocks of the input bits as the accurate part, leaving the rest as the inaccurate part. For example, a 32-bit addition, A = "1001 0011 1100 1010 1110 1010 0101 0001" and B = "1101 0011 0101 1110 1011 0111 0001 0110", the MSB block of both of the inputs are non-zero, and so the first three blocks become the accurate part leaving the last five blocks as the inaccurate part. In this case, ETAll behaves is the same as ETAl, but if the first few blocks are all "zeros", the "selector" circuit will chose its FA and SA accordingly. For example, also a 32-bit addition, A = "0000 0000 0000 1101 0101 1000 0010 1100" and B = "0000 0000 0000 1001 1110 1001 0110 0001", since the first three MSB blocks are all "zeros", ETAll will set the first six MSB blocks as their accurate part with the rest of the two blocks being the inaccurate part. The addition algorithm is explained in Figure 3.8. In this example, the inaccurate part consists of only two blocks, i.e. 8 bits and this yields a much more accurate result than ETAl.

Figure 3.8 Addition Algorithm of ETAll

3.4.2 Hardware Implementation of a 32-bit ETAlll

(A) Selector:

The design of the selector in ETAlll is an important part of circuit. There is a selector for each of the last five blocks. The main function of the selector is to select whether the Speed Adder or the Full Adder will be used for the block. The unused adder block is also required to be turned off by the selector to save power. The selector is basically an 8-bit OR gate. However, due to the input capacitance for 8 inputs will lead to a longer delay, the circuit of selector is constructed through cascading two 4-input NOR gates with a 2-input NAND gate. Inputs of the OR gates are the input number bits which are leading the selector block by three blocks as described in the Figure 3.9.

Figure 3.9 the Selector circuit in ETAlll

(B) Full Adder (FA):

The Full Adders employed in ETAlll consists of two types: a) full adders used in the first three blocks and b) full adders deployed in the remaining five blocks. Although all of them are conventional full adders, the two adder types have some differences. Type A adder are connected to Vdd directly, but not so far type B adder which is

instead controlled by the selectors. If speed adder is used in one block, the full adder in the same block will be switched off by the selector to save power.

(C) Speed Adder (SA):

Speed Adders exists only in the last five blocks. The circuit of the speed adder is given in Figure 3.10.

Figure 3.10 Circuit of Speed Adder (SA)

From Figure 3.10, it can be seen that the OR gate is controlled by not just a control signal but also selector. The OR gate is also not connected directly to the Vdd. So unless the control signal and the selector signal are both enabled, the OR gate can be connected to the Vdd and perform the calculation. Otherwise, the circuit will be switched off to save power.

Since ETAlII is just an enhancement version of ETAl, the other parts of the circuit are similar to ETAl [30].

3.4.3 Accuracy of ETAlII

As mentioned in subsection 3.2.4, ETAl have the problem of calculating small input numbers. The AP of ETAlII for different input ranges is tabulated in Table 3.10.

Table 3.10 AP of ETAlII with different MAAs & Input Ranges

Input range	$0 \sim 2^8 - 1$	$0 \sim 2^{16} - 1$	$0 \sim 2^{24} - 1$	$0 \sim 2^{32} - 1$
MAA=100%	1.0	0.7658	0.6578	0.6021
MAA=99%	1.0	1.0	1.0	1.0
MAA=98%	1.0	1.0	1.0	1.0
MAA=97%	1.0	1.0	1.0	1.0
MAA=96%	1.0	1.0	1.0	1.0
MAA=95%	1.0	1.0	1.0	1.0
MAA=94%	1.0	1.0	1.0	1.0
MAA=93%	1.0	1.0	1.0	1.0
MAA=92%	1.0	1.0	1.0	1.0
MAA=91%	1.0	1.0	1.0	1.0
MAA=90%	1.0	1.0	1.0	1.0

Table 3.10 clearly shows that the accuracy is very good for each range of input numbers is not confined in the small number or the large number range. It is a big improvement compared with ETAl and ETAlI.

3.4.4 Simulation results of ETAlII

The simulation is carried out using Chartered Semiconductor Manufacturing Ltd's 0.18- μ m CMOS process. 100 random 32-bit input sets generated by C program are used to calculate the average power at 100 MHz. Comparison of circuit performance (i.e., the power consumption, delay time, PDP, and transistor count) between the ETAlII and ETAl, ETAlI, conventional adders was been performed and the results of which are tabulated in Table 3.11. For the HSPICE code of simulation, please refer to Appendix C for the HSPICE Code of ETAlII.

Table 3.11 Simulation results for ETAI and conventional adders

Type	Power(mw)	Delay(ns)	PDP(pJ)	Transistor Count
RCA	0.22	4.04	0.89	896
CSK	0.46	2.90	1.33	1728
CSL	0.60	3.06	1.84	2176
CLS	0.51	2.37	1.21	2208
ETAI	0.13	2.29	0.30	1006
ETAI	0.24	0.85	0.20	1372
ETAIIM	0.24	1.39	0.33	1372
ETAI	0.18	2.16	0.39	1622

As seen from the simulation results, ETAI has better performance than most of the conventional adders in terms of power, delay and transistor count. It achieves very small power consumption, even lower than those seen in RCA and ETAI. The speed performance is almost the same as ETAI, which is indeed quite good. The only problem is that the circuit is a little large (1622 transistors in total). This is due to the additional “selector” circuit and the duplication of the FAs and SAs in the last five blocks.

3.5 ETA Type IV

ETAI has a problem of calculating small input numbers, ETAI has a lower accuracy compared to ETAI, ETAIIM has longer delay and ETAI has a large area. To overcome these problems, a new Probabilistic Adder for Error Tolerant Application – ETA Type IV (ETAI for short) is developed.

3.5.1 Addition Arithmetic for ETAIV

In the worst case of binary addition, the carry output will continuously flow from the LSB position to the MSB position such as 1111 1111 + 0000 0001. If the worst case

occurs, tremendous amount of time and power will be consumed along the carry propagation path. But as discussed in section 3.3.1, this worst case seldom happens indeed. In most of the cases, the carry input signal to certain bit position can be just determined by several neighboring bits which are to the right side of the concerned bit. Assume that the two input operands are perfectly random. The minimum probability of getting correct carry signal on any bit in determining the carry signal can be derived. Refer to Table 3.3 in section 3.3.1, It can be seen that when considering only one neighboring bit position, in at least 50% of the cases, one can get a correct carry signal in the following addition; if the number of bit positions to be considered is increased from 1 bit to 6 bits, the probability of getting wrong result is less than 2%; if number of bit positions considered is 8 or above, the probability of getting wrong signal is less than 0.5%.

The greater the bit positions considered, the higher the chances of securing a correct carry signal. This of course necessitates additional computation time for completing the addition. ETAIV depicted in Figure 3.11 detaches the tradeoff between the accuracy and the delay in a very nice way. ETAIV is derived from Carry Select Adder (CSA) [2]. The difference between ETAIV and CSA is that we use a 2-to-1 multiplexer (MUX2) is deployed to break the carry chain into two stages. As shown in Figure 3.12, an N-bit adder is divided into N/X blocks (each Sum Generator block has X bits). In the actual design however, N may not be exactly divided by X. Both Carry Generator Type I and II have the same X bits and perform their functions simultaneously, calculating the carry outputs with X bits accuracy.

The role of MUX2 is to select the output signal from the two Type II Carry Generators that has either logic '1' or '0' input. When the output of Carry Generator Type I is at logic '1', the output signal of Carry Generator Type II that has a logic input is selected and transferred to the most significant sum generator block through MUX2. If the output of Carry Generator Type I is 0 (logic low), the carry output of the other Carry Generator Type II with logic '0' input will be inputted into the most significant sum generator block. In this way, the accuracy of the carry input brought to the most significant sum block will be increased by two-folds without degrading the speed performance of the ETA. ETAIV [33] can therefore perfectly eliminate the drawbacks

of ETAIIM.

Figure 3.11 Architecture of ETAIV

3.5.2 Dividing strategy for ETAIV

To implement ETAIV, the block size X to be chosen must be such that the performance is within the specifications, i.e. it has to satisfy AP , MAA , etc. If X is too large, there will be a huge wastage of power and chip area. Speed performance will also be low even though the acceptance probability is very high. In real application, the delay path should be minimized by carefully choosing the sizes of the sum and carry generators. Take a 32-bit ETAIV for example, the simulation results of AP under different strategies and MAA 's are provided in Table 3.12.

Table 3.12 AP for 32-bit ETAIV under different dividing strategies and MAA's

No. of Bits MAA (%)	X = 1	X = 2	X = 4	X = 8	X = 16
100	0.0430	0.4136	0.9136	0.9985	1.0
99	0.6466	0.8848	0.9917	1.0	1.0
98	0.6695	0.9063	0.9966	1.0	1.0
97	0.6749	0.9127	0.9976	1.0	1.0
96	0.6954	0.9156	0.9981	1.0	1.0
95	0.7101	0.9214	0.9981	1.0	1.0
94	0.7155	0.9219	0.9990	1.0	1.0
93	0.7232	0.9234	0.9995	1.0	1.0
92	0.7262	0.9239	0.9995	1.0	1.0
91	0.7350	0.9239	0.9995	1.0	1.0
90	0.7374	0.9249	1.0	1.0	1.0

Compare Table 3.12 with Table 3.4 (AP for 32-bit ETAlI), we can see the probabilities of getting correct carry signal have been improved by ETAIV as compared to ETAlI.

3.5.3 Implementation of a 32-bit ETAIV

The hardware implementation of the ETAIV used the conventional designs as ETAlI. The “sum-generator” blocks in Figure 3.11 are all built by cascading four conventional 28-transistor full adders, as illustrated in Figure 2.3(a). The “carry generator” block of Figure 3.12 is implemented using the carry generator of a 4-bit carry-look-ahead adder. A 4-bit carry generator circuit is presented in Figure 2.10. The 2-to-1 multiplexer is formed by two transmission gates as shown in Figure 3.12.

Figure 3.12 Implementation of a 2-to-1 multiplexer

3.5.4 Accuracy of ETAIV

In the previous design, the main problem of ETAI is that its accuracy is poor when calculating small input numbers although it is very good in terms of power and speed performance. To overcome this problem, we designed ETAIL and ETAILI. Let's now check whether ETAIV has such problem. The simulations were carried out using C program and 10,000 random input patterns in the range of $0 \sim 2^{32} - 1$ were tested. For comparison, the AP simulation results of ETAIL are also given in Table 3.13.

Table 3.13 AP of ETAIV with different MAA's & input ranges

Input range	$0 \sim 2^8 - 1$	$0 \sim 2^{16} - 1$	$0 \sim 2^{24} - 1$	$0 \sim 2^{32} - 1$
MAA=100%	0.9688	0.9374	0.9151	0.9190
MAA=99%	0.9688	0.9673	0.9694	0.9920
MAA=98%	0.9688	0.9689	0.9706	0.9964
MAA=97%	0.9688	0.9692	0.9706	0.9973
MAA=96%	0.9688	0.9692	0.9706	0.9983
MAA=95%	0.9688	0.9692	0.9706	0.9987
MAA=94%	0.9688	0.9692	0.9706	0.9987
MAA=93%	0.9688	0.9692	0.9706	0.9987
MAA=92%	0.9688	0.9692	0.9706	0.9987
MAA=91%	0.9688	0.9692	0.9706	0.9987
MAA=90%	0.9688	0.9692	0.9706	1.0

Comparing Table 3.13 with Table 3.2 (AP of ETAI for different input ranges), it is obvious that ETAIV has much better accuracy performance a lot than ETAI. The AP values of ETAIV are also noted to be distributed evenly for different input range.

3.5.5 Simulation Results of ETAIV

The simulation is carried out using Chartered Semiconductor Manufacturing Ltd's 0.18- μm CMOS process. 100 random 32-bit input sets generated by C program are used to calculate the average power at 100MHz. The comparison of circuit performance (i.e., the power consumption, delay time, PDP, and transistor count) between the ETAIV and ETAI, ETAlI, conventional adders are shown via Table 3.14. For the HSPICE code of simulation, please refer to Appendix D – HSPICE Code of ETAIV.

Table 3.14 Simulation results for ETAs and conventional adders

Type	Power(mw)	Delay(ns)	PDP(pJ)	Transistor Count
RCA	0.22	4.04	0.89	896
CSK	0.46	2.90	1.33	1728
CSL	0.60	3.06	1.84	2176
CLS	0.51	2.37	1.21	2208
ETAI	0.13	2.29	0.30	1006
ETAIL	0.24	0.85	0.20	1372
ETAIIM	0.24	1.39	0.33	1372
ETAIIL	0.18	2.16	0.39	1622
ETAIIV	0.25	1.03	0.26	1444

As seen from the simulation results, ETAIV has better performance than most of the conventional adders in terms of power, delay and transistor count. Same as ETAIL, ETAIV can still solve the small inputs problems of ETAI with little power and delay increment. With about 5% more transistors, ETAIV provides better accuracy performance than ETAIL and shorter delay than ETAIIM.

3.6 Summary

In this Chapter, the frequently used terms for the proposed probabilistic adder circuits are first defined. Next, the Probabilistic Adder for Error-Tolerant Adder Type I (ETAI), Probabilistic Adder for Error-Tolerant Adder Type II (ETAIL), Probabilistic Adder for Error-Tolerant Adder Type III (ETAIIL), and lastly, the Probabilistic Adder for Error-Tolerant Adder Type IV (ETAIIV) are introduced. ETAI adopts the divide and conquer technique, where it calculates the accurate part and inaccurate part separately. ETAIL separates the inputs into several blocks and calculate each block at the same time. ETAIIL also adopts the divide and conquer technique but the dividing methodology is not fixed but depends on the input values. ETAIV is similar to ETAIL except that it uses the Carry Select Adder algorithm instead of the Carry Lookahead algorithm to improve the speed. The simulation results for each type of ETAs are listed in Table 3.14. And the summarization of the features and performance of the ETAs are listed in Table 3.15.

Table 3.15 Features and Performance of ETAs

Type	Feature	Performance
RCA	Carry chain from LSB to MSB	High power, high delay
ETAI	Accurate part and inaccurate part processed at same time	High speed but not good at calculating small numbers
ETAIL	Divide inputs into blocks and calculated at same time	High speed but not good at calculating large numbers
ETAIIL	Similar to ETAI but dynamically dividing the accurate part and inaccurate part	High speed and high accuracy but a little higher at PDP
ETAILV	Similar to ETAIL but uses the carry select adder structure	High speed and higher accuracy than ETAIL

Chapter 4: Applications of Probabilistic Adder Circuits

After introducing the algorithms, mathematical properties, and schematic implementations, the question naturally falls onto the actual application of the proposed probabilistic adder (ETA). The proposed probabilistic adder circuits have shown their outstanding performance in both power and speed, but the most important concerned is the practicality of these probabilistic adders and their merits.

Of course, not all digital systems can deploy the probabilistic adder circuits. In some digital systems, such as the control system, the correctness of the output signals is extremely important and hence eliminating the use of probabilistic adder circuits. However, for many digital signal processing (DSP) systems that process the signals relating to human senses such as sound, images, smells and touch, for example, the image processing and speech processing systems, the probabilistic adder circuits may be applicable. That is because human behaviors/feelings are always error-tolerant. One possible application of the error-tolerant DSP system is the medical sensing chips implanted into human bodies to help those suffering from the loss of vision or hearing. In this case, replacing batteries may require invasive surgery and thus, it is of paramount importance to prolong battery life. The ultra-low power character of the proposed designs offers a great hope to realize this. Another application is the portable electronic devices, such as cell phone and laptop, where both low-power and high-speed are desired by users as well as merits.

In the following sections, the application of ETA in DSP systems is discussed. The proposed ETA's will be used to implement the Fast Fourier Transform (FFT) function and the ETA-based FFT function will be employed in digital sound processing and image processing to demonstrate its functionality.

4.1 Digital Sound Processing

Sound signal can be stored as an array – the sound amplitude is the data value and the timing between two continuous points is the sampling interval, as illustrated in Figure 4.1. Both the Fast Fourier Transformation (FFT) and Inverse Fast Fourier

Transformation (IFFT) require adder circuits. By performing FFT followed by IFFT on sound, the results of sound processing can be evaluated.

Figure 4.1 Sound information of pronouncing “One-Two”

FFT transforms the time-domain information of the sound into frequency domain and IFFT transforms it back from frequency domain to time domain. Ideally, if FFT has been applied to the sound followed by IFFT, the sound information should not change. The computational processes of FFT and IFFT involve large number of additions and multiplications; it is therefore a good platform for embedding proposed ETAs.

The experiment was designed as follows. The sound produced due to the pronunciation of “One-Two” was first split into several blocks, where each blocks was processed by FFT and IFFT, and the resultant blocks were then combined to derive the original sound. Standard additions using conventional adder such as the RCA had been deployed and the result obtained is presented in Figure 4.2. We then replaced all the common additions of both the FFT and IFFT using our proposed addition arithmetic and repeated the simulation. Again, the sound produced due to the pronunciation of “One-Two” was divided into several blocks and each block is processed with FFT followed by IFFT but the ETAs were instead embedded. Figure 4.3 shows the result with ETAlll embedded.

Figure 4.2 Sound Signal after FFT and IFFT using conventional addition arithmetic.

Figure 4.3 Sound Signal after FFT and IFFT, with ETAlll arithmetic embedded.

From Figure 4.2 and Figure 4.3, we can see that there are some differences and this is due to the error of the ETAlll's arithmetic logic. Nevertheless, the resultant sounds were hardly differentiable by human ears. For numerical comparison, with MAA set to 95%, the AP of the array shown in Figure 4.3 is 99.3% as compared to that of Figure 4.2.

To be more specific, during FFT on the sound information bit array, the real number sound bits are firstly transferred into a complex number by multiplying e to some value (refer to appendix E). And these transferred complex numbers will be summed

up with two loops to produce the output array. During the summing up for the complex number, the real part and image part is calculated separately and the ETAlll will be used to sum up these values (refer to appendix F) to produce the frequency domain output. For the IFFT, the similar way will be performed to obtain the time domain sound information.

Another way to compare the sound information between the sounds transferred with normal FFT and the transferred with FFT embedded ETAs is to compare their Power Spectral Density (PSD). The Power Spectral Density is a positive real function of a frequency variable associated with a stationary stochastic process, or a deterministic function of time, which has dimensions of power per Hz, or energy per Hz. It is often called simply the *spectrum* of the signal. Intuitively, the spectral density captures the frequency content of a stochastic process and helps identify periodicities. In physics, the spectral density of the sound wave, when multiplied by an appropriate factor, will give the power carried by the wave, known as the Power Spectral Density of the signal.

Figure 4.4 shows the PSD of the original sound information pronouncing “One-Two”, and Figure 4.5 and 4.6 present the PSD of the sound information transferred with normal FFT and ETAlll embedded FFT, respectively. From Figure 4.5 and 4.6, there are negligible differences between PSD of sound after FFT and IFFT with conventional addition algorithm and PSD of sound after FFT and IFFT with ETAlll addition algorithm. For numerical comparison, with MAA set to 95%, the AP of the array shown in Figure 4.6 is 99.1% as compared to that of Figure 4.5.

Figure 4.4 PSD of Sound information of pronouncing "One-Two"

Figure 4.5 PSD of Sound information after FFT and IFFT with conventional addition algorithm

**Figure 4.6 PSD of Sound information after FFT and IFFT with ETAlII algorithm
embedded**

4.2 Digital Image Processing

Visual information plays an important role in almost every area of people's life. Today, much of this information is represented and processed digitally.

A digital image is stored as an array in computers. The value of each element represents the color of the corresponding pixel. For a grayscale image, the image is represented by a 2-dimensional (m-by-n) array. If the array is *double* (i.e., the data type of the elements in this array is *double*), the pixel value of 0.0 represents the color of black, the pixel value of 1.0 represents white, and the pixel values in between represent the different levels of gray; if the array is *uint8* (unsigned 8-bit integer), the white is represented by the pixel value of 255; if the array is *uint16* (unsigned 16-bit integer), the pixel value of 65535 represents the color of white. For convenience, in the rest of this thesis, if no notification is given, the image arrays to be discussed are all assumed to be of *uint8* arrays. For a color image, the image is represented by a 3-dimensional (m-by-n-3) array. The value for each pixel consists of

three numbers giving the decomposition of the color in the three primary colors Red (R), Green (G), and Blue (B). The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white is coded as $R = 255$, $G = 255$, and $B = 255$; black will be known as $(R, G, B) = (0, 0, 0)$; and say, bright pink will be $(255, 0, 255)$. That is why the true-color image is also called RGB image. A grayscale image can be converted to true-color format by concatenating three copies of the original matrix along the third dimension. Here, we focus mainly on the grayscale image.

Because each digital grayscale image is represented by an m -by- n matrix, the Fourier transform towards an image is actually performed on its representing matrix. After the transform, the *uint8* matrix will become a complex matrix whose elements are complex numbers. If inverse Fourier transform is performed on this complex matrix, the original matrix will be retrieved.

Figure 4.4 presents a very simple digital image, which is only a white spot in a black background, and its corresponding 16-by-16 matrix **A**. To demonstrate the functions of FFT and inverse FFT that have been introduced in previous sections, the standard FFT is first performed on matrix **A**, after which follows the inverse FFT. The simulation results are shown in Figure 4.5. From the figure, it can be seen that after FFT, the original matrix **A** has been transformed to a complex matrix, which can be regarded as the frequency domain representation of matrix **A**. The inverse FFT that is performed thereafter on the obtained complex matrix transforms it back to the original matrix **A**.

(a) The image of a white dot in a black background

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) The 16-by-16 matrix representing the image

Figure 4.7 A simple digital image and its matrix representation

Figure 4.8 Matrix A after FFT and inverse FFT

With the knowledge presented above, the following experiment is devised to validate the usability of the ETAs. An image was first translated to a matrix and sent through a standard transfer system that used normal FFT and normal reverse FFT. The matrix output of this system was then transformed back to an image and presented in Figure 4.6(a). The matrix of the same image was also processed in a system that used the inaccurate FFT and inaccurate reverse FFT, where both FFTs had incorporated the 32-bit ETAI, with the processed image given in Figure 4.6(b). Figure 4.6 (c) and 4.6 (d) are the images processed with FFT and IFFT, with ETAlI and ETAIV embedded separately.

From these resultant pictures of Figures 4.6, we can see some differences from the different pictures. With a MAA setting of 95%, the AP of the matrix representation of Figure 4.6(b) is 98.3% as compared to the matrix representation of Figure 4.6(a); the AP of the matrix representation of Figure 4.6(c) is 89.5% as compared to the matrix representation of Figure 4.6(a); and the AP of matrix representation of Figure 4.6(d) is 99.2% as compared to the matrix representation of Figure 4.6(a).

(a)

(b)

(c)

(d)

Figure 4.9 Images after FFT and inverse FFT: (a) image processed with conventional adder and (b) image processed with the ETAI (c) image processed with ETAlI (d) image processed with ETAIV.

4.3 Summary

This Chapter illustrates the viability of the proposed circuits through various applications that could embed the proposed Probabilistic Adder for Error-Tolerant Application (ETAs). The proposed designs are suitable for any applications that can tolerate some amount of errors which in most cases are applications that usually relate to human senses, e.g., visual and sound. In this thesis, both the sound and image processing had been chosen to embed the proposed ETAs. The results showed that ETAs are suitable for these applications. While losing some accuracy, the improvement in the PDP (Power Delay Product) and speed are substantial making the concept extremely interesting and in fact, wise.

Chapter 5: Conclusion and Future work

5.1 Conclusion

This project has adopted an emerging concept in digital integrated circuit design and testing, the *Error Tolerance*, in the design of digital parallel adder to propose a new type of adder, named the *Probabilistic Adder for Error Tolerant Application (ETA)*. Both the mathematical properties and electronic performance of these new types of adder have been thoroughly studied and investigated.

Several different implementations of the enhanced ETA have been proposed, namely the ETA Type II (ETAII), ETA Type III (ETAIII) and ETA Type IV (ETAIV).

The ETAII is based on the idea that in most cases the carry signal for a bit position is determined by several neighboring bits instead of all the bits on its right. Hence, the critical path of the whole circuit can be greatly curtailed by dividing the whole adder into a number of blocks and conducting the addition operations in each block concurrently. With this method, improvements in power consumption and speed performance can be achieved. Simulation results show that the PDP of ETAIIM (a modified structure of the original ETAII) is 60%, 75%, 81%, and 71% better than that of the RCA, CSK, CSL, and CLA, respectively.

The basic idea of ETAIII is to combine the advantages of ETAI and ETAII. It adopts the idea of having “accurate part – inaccurate part” mode of ETAI and also the “block separation mode” of ETAII. This allows ETAIII to achieve better accuracy for all ranges of the inputs (the AP is evenly distributed in different range of inputs and is good for both large inputs and small input numbers). The simulation results showed that the PDP of ETAIII is also far better than the conventional adders but comparable to that of ETAI and ETAII. In terms of accuracy, ETAIII is much better than both ETAI and ETAII. The trade off

is that it consumes a large area (transistor count) due to the duplicate FAs/SAs in the inaccurate part.

ETAIV is based on the concept of “block dividing”, as in ETAll, but different from ETAll, it adopts the architecture of “Carry-Select-Adder” instead of “Carry-Lookahead-Adder”. The 2-to-1 multiplexer used in the circuit helps to derive better accuracy while maintaining its fast speed. The simulation results show indicated ETAIV to have very small PDP, the same as the of ETA series, which is far better than the conventional adders. Although its area (transistor count) is a bit larger than the previous ETAs, its accuracy is better.

The ultra-low power and high speed characteristics of the ETAs provide an innovative way to break through the bottleneck of conventional digital IC design technology that always hinders the circuit designers from achieving both the low-power and high-speed simultaneously. When the accuracy of a circuit is brought into the design process, the two dimensional trade-off between power and speed becomes three dimensional, i.e., power-speed-accuracy. Adding the new design parameter provides the circuit designers with more flexibility when designing a circuit. If low-power and high-speed are desired, the accuracy can be used as a trade-off, and vice versa.

As described, ETA circuits demonstrate a bright future of the error-tolerant applications. In the near future, the error-tolerant circuit/system may become an important part in many electronic devices, such as the biomedical electronic devices and personal portable electronic devices.

5.2 Future Work

This project has started a new direction of digital IC and system design. On the road of exploring in this new direction, there are a lot more work that can be and need to be done.

Firstly, other implementations of ETAs other than the proposed ETAlI, ETAlII and ETAlV in this thesis can be investigated. The aim of designing an ETA is to find a structure that can gain significant improvements in power consumption and speed performance while still maintaining a high accuracy (as high as possible). An efficient ETA structure can trade only a little loss in accuracy for great improvements in other metrics.

Secondly, the probabilistic circuit for error tolerant application can be extended from adder to other digital circuits and eventually the whole digital system. The concept of ETA has been proven to be feasible by employing it in digital image processing. It is therefore logical to extend this concept to other digital circuits such as the multiplier, and to a whole DSP system.

Thirdly, the applications of the error-tolerant circuit/system can be studied and investigated, such as sound application, image application etc. All the applications related to the human sense can adopt the error-tolerant concept.

Lastly, the error-tolerant design can be extended from the Application Specific Integrated Circuit (ASIC) implementation to the Field Programmable Gate Array (FPGA) implementation. FPGA is becoming more and more popular in digital system design because of its many advantages over the ASIC. Hence, employing the concept of Error-Tolerance in FPGA implementation may also be a valuable work.

List of Publications

- [1] Zhu Ning, Zhang Weijia, Goh Wang Ling, Yeo Kiat Seng, and Kong Zhi Hui, “Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 18, Issue 8, pp. 1225–1229, Aug 2010.
- [2] Zhu Ning, Goh Wang Ling, Yeo, Kiat Seng, “An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application”, *International Symposium on Integrated Circuits (ISIC 2009)*, Singapore, pp. 69-72, 14 to 16 Dec 2009.
- [3] Zhu Ning, Wang Ling Goh, Wang Gang and Yeo Kiat Seng, “Enhanced Low-Power High-Speed Adder for Error-Tolerant Application”, accepted for *7th International SoC Design Conference (ISOCC 2010)*, Nov 22-23.
- [4] Zhu Ning, Wang Ling Goh, and Yeo Kiat Seng, “Ultra Low-Power High-Speed Flexible Truncated Adder for Error-Tolerant Applications”, to be submitted to *9th IEEE international NEWCAS Conference (NEWCAS 2011)*, Bordeaux, France, June 26-29, 2011.

Reference

- [1] Gordon E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, 1965.
- [2] K. L. Shepard, "Design methodologies for noise in digital integrated circuits," in *Proc. Design Automation Conference*, pages 94-99, San Francisco, California, Jun 1998.
- [3] L. B. Kish, "End of Moore's law: thermal (noise) death of integration in micro and nano electronics," *Physics Letters A*, 305: 144-149, Dec 2002.
- [4] K. Natori and N. Sano, "Scaling limit of digital circuits due to thermal noise," *Journal of Applied Physics*, 83: 5019--5024, May 1998.
- [5] R. Hegde and N. R. Shanbhag, "Toward achieving energy efficiency in presence of deep submicron noise," *IEEE Trans. VLSI Syst.*, 8: 379-391, Aug 2000.
- [6] J. D. Meindl, "Low power microelectronics: retrospect and prospect," *Proc. IEEE*, 83: 619-635, Apr 1995.
- [7] <http://public.itrs.net/Files/2003ITRS/Home2003.htm>
- [8] Paul H. Bardell, *Built-in Test for VLSI*, Wiley, 1987.
- [9] N. R. Shanbhag, "Reliable and efficient system-on-chip design," *IEEE Computer*, 37(3): 42-50, May 2004.
- [10] Melvin A. Breuer, "Let's think analog," *Proc. of the IEEE Computer Society Annual Symposium on VLSI*, 2005.
- [11] M. A. Breuer, "Intelligible test techniques to support error-tolerance," *Proc. Asian Test Symp.*, pp. 386--393, 2004.
- [12] Kuen-Jong Lee, Tong-Yu Hsieh and M. A. Breuer, "A novel test methodology based on error-rate to support error-tolerance," *Proc. International Test Conf.*, pp. 1136-1144, 2005.
- [13] Tong-Yu Hsieh, Kuen-Jong Lee, and Melvin A. Breuer, "Reduction of detected acceptable faults for yield improvement via error-tolerance," *Design, Automation and Test in Europe Conference and Exhibition*, pp. 1-6, Apr 2007.

- [14] M. A. Breuer, S. K. Gupta, and T. M. Mak, "Defect and error-tolerance in the presence of massive numbers of defects," *IEEE Design and Test of Computers* 21(3): 216-227, 2004.
- [15] Melvin A. Breuer and Haiyang Zhu, "Error-tolerance and multi-media," in *Proc. of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2006.
- [16] *Recommendation ITU-T P.830, Subjective performance assessment of telephone band and wideband digital codecs*, Int'l Telecommunications Union, 1996.
- [17] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, *Digital Integrated Circuits*, Prentice Hall, 2003.
- [18] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A System Perspective*. Reading, MA: Addison-Wesley, 1993.
- [19] M. Lehman and N. Burla, "Skip techniques for high-speed carry propagation in binary arithmetic units," *IRE Transactions on Electronic Computers*, vol.EC-10, pp.691-698, Dec 1962.
- [20] O. Bedrij, "Carry Select Adder," *IRE Trans. on Electronic Computers*, vol.EC-11, pp.340--346, 1962.
- [21] O. MacSorley, "High speed arithmetic in binary computers," *IRE Proceedings*, vol.49, pp.67-91, 1961.
- [22] Wallace, W.F, Dlay, S.S, Hinton, O.R, "Probabilistic carry state estimate for improved asynchronous adder performance", *Computers and Digital Techniques, IEE proceedings*, page 221-226, 2001.
- [23] Ashmila, E.M, Dlay, S.S, Hinton, O.R, "Adder methodology and design using probabilistic multiple carry estimates", *Computers and Digital Techniques, IEE proceedings*, page 697-703, 2005.
- [24] Lau, M.S.K, Ling, K.V, Yun Chung Chun, Bhanu, A, "Modeling of Probabilistic Ripple-Carry Adders", *Electronic Design, Test and Application*, page 201-206, 2010.
- [25] N. R. Shanbhag, "Reliable and efficient system-on-chip design," *IEEE Computer*, 37(3): 42-50, May 2004.

- [26] Melvin A. Breuer, "Let's think analog," *Proc. of the IEEE Computer Society Annual Symposium on VLSI*, pp.2-5, 2005.
- [27] <http://public.itrs.net/Files/2003ITRS/Home2003.htm>
- [28] M. A. Breuer, S. K. Gupta, and T. M. Mak, "Defect and error-tolerance in the presence of massive numbers of defects," *IEEE Design and Test of Computers* 21(3): 216-227, 2004.
- [29] Kuen-Jong Lee, Tong-Yu Hsigh and M. A. Breuer, "A novel test methodology based on error-rate to support error-tolerance," *Proc. International Test Conf.*, pp. 1136-1144, 2005.
- [30] Zhu Ning, Zhang Weijia, Goh Wang Ling, Yeo Kiat Seng, and Kong Zhi Hui, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 18, Issue 8, pp. 1225–1229, Aug 2010.
- [31] Zhu Ning, Goh Wang Ling, Yeo, Kiat Seng, "An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application", *International Symposium on Integrated Circuits (ISIC 2009)*, Singapore, pp. 69-72. 14 to 16 Dec 2009.
- [32] Zhu Ning, Wang Ling Goh, and Yeo Kiat Seng, "Ultra Low-Power High-Speed Flexible Truncated Adder for Error-Tolerant Applications", to be submit to 9th *IEEE international NEWCAS Conference (NEWCAS 2011)*, Bordeaux, France, Jun 26-29, 2011.
- [33] Zhu Ning, Wang Ling Goh, Wang Gang and Yeo Kiat Seng, "Enhanced Low-Power High-Speed Adder for Error-Tolerant Application", accept for 7th *International SoC Design Conference (ISOCC 2010)*, Nov 22-23.
- [34] K. V. Palem, "Energy aware computing through probabilistic switching: a study of limits," *IEEE Trans. on Computers*, Vol. 54, No. 9, Sep 2005.
- [35] S. Cheemalavagu, P. Korkmaz, and K. V. Palem, "Ultra low energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," In *Proc. of the 2004 International Conference on Solid State Devices and Materials*, pages 402-403, Tokyo, Japan, Sep 2004.

- [36] P. Korkmaz, B. E. S. Akgul, K. V. Palem Palem and L. N. Chakrapani, "Advocating noise as an agent for ultra-low energy computing: probabilistic complementary metal-oxide-semiconductor devices and their characteristics," *Japanese Journal of Applied Physics*, Vol. 45, No. 4B, 2006.
- [37] C. Oliver and S. Quegan, *Understanding Synthetic Radar Images*. Raleigh, NC: SciTch Publishing, 2004.
- [38] I. Chong and A. Ortega, "Hardware testing for error tolerant multimedia compression based on linear transforms," *Proc. International Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 523-531, 2005.
- [39] N. Takagi, H. Tago, C. R. Baugh, and S. MUroga, *Logic Design*, CRC Press, 2003.
- [40] Paul H. Bardell, *Built-in Test for VLSI*, Wiley, 1987.
- [41] Kiat-Seng Yeo, Kaushik Roy, *Low-Voltage, Low-Power VLSI Subsystems*, McGraw-Hill, 2005.
- [42] E. O. Brigham, *The fast Fourier transform and its applications*, Prentice Hall, 1988.
- [43] James W. Cooley, and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.* 19: 297-301, 1965.
- [44] Georg Bruun, "z-Transform DFT filters and FFTs," *IEEE Trans. on Acoustics, Speech and Signal Processing (ASSP)* 26 (1), 56-63, 1978.
- [45] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE* 56, 1107-1108, 1968.
- [46] Leo I. Bluestein, "A linear filtering approach to the computation of the discrete Fourier transform," *Northeast Electronics Research and Engineering Meeting Record* 10, 218-219, 1968.
- [47] John G. Proakis, and Dimitris Manolakis, *Digital Signal Processing - Principles, Algorithms and Applications*, Pearson.
- [48] Rafael C. Gonzales, *Digital Image Processing*, Prentice Hall, 2002.
- [49] Gerard Blanchet, *Digital Signal and Image Processing Using Matlab*, ISTE Ltd., 2006.

Appendix

Appendix A – C Code of calculation AP for ETail

```
#####
# This code is to calculate the AP for ETail
# Author: Zhu Ning
# Date: 2008-12-09
#####
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>

#define TOTAL_NUMBER 32
#define BLOCK_SIZE 4
#define BLOCK_LOW 5

#define SAMPLES 10000
#define th0 1.0
#define th1 0.99
#define th2 0.98
#define th3 0.97
#define th4 0.96
#define th5 0.95
#define th6 0.94
#define th7 0.93
#define th8 0.92
#define th9 0.91
#define th10 0.90

int carry(int A, int B, int CIN)
{
    int COUT;
    if((A+B+CIN)>(pow(2,BLOCK_SIZE)-1)) COUT=1;
    else COUT=0;
    return COUT;
}

int sum(int A, int B, int CIN)
{
    int S;
    S=A+B+CIN;
    if(S>(pow(2,BLOCK_SIZE)-1)) S=S-pow(2,BLOCK_SIZE);
    return S;
}

void main()
{
    int i=0, j=0, t=0, ground=0, n0=0,n1=0,n2=0,n3=0,n4=0,n5=0,n6=0,n7=0,n8=0,n9=0,n10=0;
    long temp_A=0, temp_B=0, obt_result=0, cor_result=0;
    char A[TOTAL_NUMBER]={0}, B[TOTAL_NUMBER]={0};
```

```

    int cell_A[TOTAL_NUMBER/BLOCK_SIZE]={0}, cell_B[TOTAL_NUMBER/BLOCK_SIZE]={0},
    cin[TOTAL_NUMBER/BLOCK_SIZE+1]={0}, cell_S[TOTAL_NUMBER/BLOCK_SIZE]={0};
    double
    accuracy,prob_0,prob_1,prob_2,prob_3,prob_4,prob_5,prob_6,prob_7,prob_8,prob_9,prob_10;
    srand(time(NULL));
    for(t=0;t<SAMPLES;t++)
    {
        temp_A=0, temp_B=0, obt_result=0, cor_result=0;
        for(i=0; i<(TOTAL_NUMBER/BLOCK_SIZE); i++)
        {
            cell_A[i]=0;
            cell_B[i]=0;
        }

        for(i=0; i<TOTAL_NUMBER; i++)
        {
            A[i]=rand()%2;
            B[i]=rand()%2;
        }
        for(i=0; i<(TOTAL_NUMBER/BLOCK_SIZE); i++)
        {
            for(j=0; j<BLOCK_SIZE; j++)
            {
                cell_A[i]=cell_A[i]+A[BLOCK_SIZE*i+j]*pow(2,j);
                cell_B[i]=cell_B[i]+B[BLOCK_SIZE*i+j]*pow(2,j);
            }
            //cin[i+1]=carry(cell_A[i], cell_B[i], ground);
        }
        for(i=0; i<BLOCK_LOW; i++)
        {
            cin[i+1]=carry(cell_A[i], cell_B[i], ground);
        }
        for(i=BLOCK_LOW; i<(TOTAL_NUMBER/BLOCK_SIZE); i++)
        {
            cin[i+1]=carry(cell_A[i], cell_B[i], cin[i]);
        }

        for(i=0; i<(TOTAL_NUMBER/BLOCK_SIZE); i++)
        {
            cell_S[i]=sum(cell_A[i], cell_B[i], cin[i]);
            obt_result=obt_result+cell_S[i]*pow(2,BLOCK_SIZE*i);
        }
        obt_result=obt_result+cin[TOTAL_NUMBER/BLOCK_SIZE]*pow(2,TOTAL_NUMBER);
        for(i=0; i<TOTAL_NUMBER; i++)
        {
            temp_A=temp_A+A[i]*pow(2,i);
            temp_B=temp_B+B[i]*pow(2,i);
        }
        cor_result=temp_A+temp_B;
        if(cor_result==0) accuracy=1.0;
        else accuracy=(obt_result*1.0)/(cor_result*1.0);
        if(accuracy==th0) n0++;
        if(accuracy>=th1) n1++;
    }

```

```

        if(accuracy>=th2) n2++;
        if(accuracy>=th3) n3++;
        if(accuracy>=th4) n4++;
        if(accuracy>=th5) n5++;
        if(accuracy>=th6) n6++;
        if(accuracy>=th7) n7++;
        if(accuracy>=th8) n8++;
        if(accuracy>=th9) n9++;
        if(accuracy>=th10) n10++;
    }
    prob_0=n0/(SAMPLES*1.0);
    prob_1=n1/(SAMPLES*1.0);
    prob_2=n2/(SAMPLES*1.0);
    prob_3=n3/(SAMPLES*1.0);
    prob_4=n4/(SAMPLES*1.0);
    prob_5=n5/(SAMPLES*1.0);
    prob_6=n6/(SAMPLES*1.0);
    prob_7=n7/(SAMPLES*1.0);
    prob_8=n8/(SAMPLES*1.0);
    prob_9=n9/(SAMPLES*1.0);
    prob_10=n10/(SAMPLES*1.0);

    printf("The probability w.r.t %f: %f\n",th0,prob_0);
    printf("The probability w.r.t %f: %f\n",th1,prob_1);
    printf("The probability w.r.t %f: %f\n",th2,prob_2);
    printf("The probability w.r.t %f: %f\n",th3,prob_3);
    printf("The probability w.r.t %f: %f\n",th4,prob_4);
    printf("The probability w.r.t %f: %f\n",th5,prob_5);
    printf("The probability w.r.t %f: %f\n",th6,prob_6);
    printf("The probability w.r.t %f: %f\n",th7,prob_7);
    printf("The probability w.r.t %f: %f\n",th8,prob_8);
    printf("The probability w.r.t %f: %f\n",th9,prob_9);
    printf("The probability w.r.t %f: %f\n",th10,prob_10);
}

```

Appendix B – HSPICE Code of ETAIIM

```

ETAIIM
.OPTIONS NODE POST
*.VEC 'input_adder_32.dat'
.VEC 'input_adder_32_ETAIIM.dat'
.GLOBAL VDD
VDD VDD 0 VOLT
.PARAM NPL=180n NW=0.3u PW=0.6u VOLT=1.8V
*.TRAN 1n 1u
.TRAN 0.1n 20n

*****INVERTER*****
.SUBCKT INVERTER IN OUT VN VP
M1 OUT IN VP VDD pmos_1p8 L=NPL W=PW
M2 OUT IN VN 0 nmos_1p8 L=NPL W=NW
.ENDS

*****2-INPUT AND GATE*****
.SUBCKT AND2 A B OUT
M1 OUT_B A VDD VDD pmos_1p8 L=NPL W=PW
M2 OUT_B B VDD VDD pmos_1p8 L=NPL W=PW
M3 OUT_B A TEMP 0 nmos_1p8 L=NPL W=0.6U
M4 TEMP B 0 0 nmos_1p8 L=NPL W=0.6U
X1 OUT_B OUT 0 VDD INVERTER
.ENDS

*****2-INPUT OR GATE*****
.SUBCKT OR2 A B OUT
M1 TEMP A VDD VDD pmos_1p8 L=NPL W=1.2U
M2 OUT_B B TEMP VDD pmos_1p8 L=NPL W=1.2U
M3 OUT_B A 0 0 nmos_1p8 L=NPL W=NW
M4 OUT_B B 0 0 nmos_1p8 L=NPL W=NW
X1 OUT_B OUT 0 VDD INVERTER
.ENDS

*****FOUR-BIT CLA LOGIC*****
.SUBCKT CLA_LOGIC CIN G0 G1 G2 G3 P0 P1 P2 P3 COUT
M1 PD1 G3 VDD VDD pmos_1p8 L=NPL W=PW
M2 PD2 G2 PD1 VDD pmos_1p8 L=NPL W=PW
M3 PD3 G1 PD2 VDD pmos_1p8 L=NPL W=PW
M4 PD4 G0 PD3 VDD pmos_1p8 L=NPL W=PW
M5 COUTB CIN PD4 VDD pmos_1p8 L=NPL W=PW
M6 COUTB P3 VDD VDD pmos_1p8 L=NPL W=PW
M7 COUTB P2 PD1 VDD pmos_1p8 L=NPL W=PW
M8 COUTB P1 PD2 VDD pmos_1p8 L=NPL W=PW
M9 COUTB P0 PD3 VDD pmos_1p8 L=NPL W=PW
M10 COUTB CIN ND4 0 nmos_1p8 L=NPL W=NW
M11 ND4 P0 ND3 0 nmos_1p8 L=NPL W=NW
M12 ND3 P1 ND2 0 nmos_1p8 L=NPL W=NW
M13 ND2 P2 ND1 0 nmos_1p8 L=NPL W=NW
M14 ND1 P3 0 0 nmos_1p8 L=NPL W=NW
M15 COUTB G0 ND3 0 nmos_1p8 L=NPL W=NW

```

```

M16 COUTB G1 ND2 0 nmos_1p8 L=NPL W=NW
M17 COUTB G2 ND1 0 nmos_1p8 L=NPL W=NW
M18 COUTB G3 0 0 nmos_1p8 L=NPL W=NW
X1 COUTB COUT 0 VDD INVERTER
.ENDS

.SUBCKT CLA4 C0 A3 A2 A1 A0 B3 B2 B1 B0 COUT4
X1 A0 B0 G0 AND2
X2 A1 B1 G1 AND2
X3 A2 B2 G2 AND2
X4 A3 B3 G3 AND2
X5 A0 B0 P0 OR2
X6 A1 B1 P1 OR2
X7 A2 B2 P2 OR2
X8 A3 B3 P3 OR2

X9 C0 G0 G1 G2 G3 P0 P1 P2 P3 COUT4 CLA_LOGIC

.ENDS

```

*****28-TRANSISTORS FULL ADDER*****

```

.SUBCKT FA A B CIN SUM COUT
M1 T1 A VDD VDD pmos_1p8 L=NPL W=PW
M2 T1 B VDD VDD pmos_1p8 L=NPL W=PW
M3 T2 B T1 VDD pmos_1p8 L=NPL W=PW
M4 X A T2 VDD pmos_1p8 L=NPL W=PW
M5 X CIN T1 VDD pmos_1p8 L=NPL W=PW
M6 X CIN T3 0 nmos_1p8 L=NPL W=NW
M7 T3 A 0 0 nmos_1p8 L=NPL W=NW
M8 T3 B 0 0 nmos_1p8 L=NPL W=NW
M9 X A T4 0 nmos_1p8 L=NPL W=NW
M10 T4 B 0 0 nmos_1p8 L=NPL W=NW
M11 T5 CIN VDD VDD pmos_1p8 L=NPL W=PW
M12 T5 A VDD VDD pmos_1p8 L=NPL W=PW
M13 T5 B VDD VDD pmos_1p8 L=NPL W=PW
M14 T6 A T5 VDD pmos_1p8 L=NPL W=PW
M15 T7 B T6 VDD pmos_1p8 L=NPL W=PW
M16 T8 CIN T7 VDD pmos_1p8 L=NPL W=PW
M17 T8 X T5 VDD pmos_1p8 L=NPL W=PW
M18 T8 X T9 0 nmos_1p8 L=NPL W=NW
M19 T9 A 0 0 nmos_1p8 L=NPL W=NW
M20 T9 B 0 0 nmos_1p8 L=NPL W=NW
M21 T9 CIN 0 0 nmos_1p8 L=NPL W=NW
M22 T8 CIN T10 0 nmos_1p8 L=NPL W=NW
M23 T10 A T11 0 nmos_1p8 L=NPL W=NW
M24 T11 B 0 0 nmos_1p8 L=NPL W=NW
M25 SUM T8 VDD VDD pmos_1p8 L=NPL W=PW
M26 SUM T8 0 0 nmos_1p8 L=NPL W=NW
M27 COUT X VDD VDD pmos_1p8 L=NPL W=PW
M28 COUT X 0 0 nmos_1p8 L=NPL W=NW
.ENDS

.SUBCKT RCA4 C0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4

```

```
X1 A0 B0 C0 S0 C1 FA
X2 A1 B1 C1 S1 C2 FA
X3 A2 B2 C2 S2 C3 FA
X4 A3 B3 C3 S3 C4 FA
.ENDS
```

```
.SUBCKT ADDER_CELL4_LOW C0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4
X1 0 A3 A2 A1 A0 B3 B2 B1 B0 C4 CLA4
X2 C0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4A RCA4
.ENDS
```

```
.SUBCKT ADDER_CELL4_HIGH C0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4
X1 C0 A3 A2 A1 A0 B3 B2 B1 B0 C4 CLA4
X2 C0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4A RCA4
.ENDS
```

```
X1 0 A3 A2 A1 A0 B3 B2 B1 B0 S0 S1 S2 S3 C4 ADDER_CELL4_LOW
X2 C4 A7 A6 A5 A4 B7 B6 B5 B4 S4 S5 S6 S7 C8 ADDER_CELL4_LOW
X3 C8 A11 A10 A9 A8 B11 B10 B9 B8 S8 S9 S10 S11 C12 ADDER_CELL4_LOW
X4 C12 A15 A14 A13 A12 B15 B14 B13 B12 S12 S13 S14 S15 C16 ADDER_CELL4_LOW
X5 C16 A19 A18 A17 A16 B19 B18 B17 B16 S16 S17 S18 S19 C20 ADDER_CELL4_LOW
X6 C20 A23 A22 A21 A20 B23 B22 B21 B20 S20 S21 S22 S23 C24 ADDER_CELL4_HIGH
X7 C24 A27 A26 A25 A24 B27 B26 B25 B24 S24 S25 S26 S27 C28 ADDER_CELL4_HIGH
*X8 C28 A31 A30 A29 A28 B31 B30 B29 B28 S28 S29 S30 S31 C32 ADDER_CELL4

*X6 C20 A23 A22 A21 A20 B23 B22 B21 B20 S20 S21 S22 S23 C24 ADDER_CELL4_LOW
*X7 C24 A27 A26 A25 A24 B27 B26 B25 B24 S24 S25 S26 S27 C28 ADDER_CELL4_LOW
X8 C28 A31 A30 A29 A28 B31 B30 B29 B28 S28 S29 S30 S31 C32 RCA4
```

```
*.MEASURE TRAN TOTAL_POWER AVG POWER FROM=0 TO=1u
```

```
.MEAS TRAN Tdelay TRIG V(A16) val=0.9 TD=0n RISE=1 TARG V(S31) val=0.9 FALL=1
*.MEAS TRAN Tdelay TRIG V(A24) val=0.9 TD=0n RISE=1 TARG V(S31) val=0.9 FALL=1
```

```
.LIB 'E:\documents\research\tools\workstation\HSpice_Models\sm093001-1j.hspice' typical
.END
```

Appendix C – HSPICE Code of ETAlII

ETAlII

*****NOTES*****

* This is an implementation of the 32-bit ETAlII *

.OPTIONS NODE POST

.VEC 'D:\ETA2\HSPICE files\Data\input_ETAlII_delay.dat'

*.VEC 'D:\WeiJia\hspice\Data\input_adder_32_test.dat'

*.VEC 'D:\WeiJia\hspice\Data\input_adder_32.dat'

.GLOBAL VDD

VDD VDD 0 VOLT

.PARAM NPL=180n NW=0.3u PW=0.6u VOLT=1.8V

*.TRAN 1n 1u

.TRAN 1n 40n

*****INVERTER*****

.SUBCKT INVERTER IN OUT VN VP

M1 OUT IN VP VDD pmos_1p8 L=0.18U W=0.6U

M2 OUT IN VN 0 nmos_1p8 L=0.18U W=0.3U

.ENDS

*****2-INPUT NOR GATE*****

.SUBCKT NOR2 A B OUT

M1 temp A VDD VDD pmos_1p8 L=0.18U W=1.2U

M2 OUT B temp VDD pmos_1p8 L=0.18U W=1.2U

M3 OUT A 0 0 nmos_1p8 L=0.18U W=0.3U

M4 OUT B 0 0 nmos_1p8 L=0.18U W=0.3U

.ENDS

*****2-INPUT OR GATE*****

.SUBCKT OR2 A B OUT

M1 temp A VDD VDD pmos_1p8 L=0.18U W=1.2U

M2 OUT_B B temp VDD pmos_1p8 L=0.18U W=1.2U

M3 OUT_B A 0 0 nmos_1p8 L=0.18U W=0.3U

M4 OUT_B B 0 0 nmos_1p8 L=0.18U W=0.3U

X1 OUT_B OUT 0 VDD INVERTER

.ENDS

*****2-INPUT AND GATE*****

.SUBCKT AND A B OUT

M1 OUT_B A VDD VDD pmos_1p8 L=0.18U W=0.6U

M2 OUT_B B VDD VDD pmos_1p8 L=0.18U W=0.6U

M3 OUT_B A TEMP 0 nmos_1p8 L=0.18U W=0.6U

M4 TEMP B 0 0 nmos_1p8 L=0.18U W=0.6U

X1 OUT_B OUT 0 VDD INVERTER

.ENDS

*****3-INPUT AND GATE*****

.SUBCKT AND3 A B C OUT

M1 OUT_B A VDD VDD pmos_1p8 L=0.18U W=0.6U

M2 OUT_B B VDD VDD pmos_1p8 L=0.18U W=0.6U

M3 OUT_B C VDD VDD pmos_1p8 L=0.18U W=0.6U


```
M4 OUT_B A TEMP1 0 nmos_1p8 L=0.18U W=0.9U
M5 TEMP1 B TEMP2 0 nmos_1p8 L=0.18U W=0.9U
M6 TEMP2 C 0 0 nmos_1p8 L=0.18U W=0.9U
X1 OUT_B OUT 0 VDD INVERTER
.ENDS
```

```
*****NAND GATE*****
.SUBCKT NAND A B OUT
M1 OUT A VDD VDD pmos_1p8 L=0.18U W=0.6U
M2 OUT B VDD VDD pmos_1p8 L=0.18U W=0.6U
M3 OUT A TEMP 0 nmos_1p8 L=0.18U W=0.6U
M4 TEMP B 0 0 nmos_1p8 L=0.18U W=0.6U
.ENDS
```

```
*****SPECIAL OR GATE*****
.SUBCKT ORS A B OUT VN VP
M1 TEMP A VP VDD pmos_1p8 L=0.18U W=1.2U
M2 OUT_B B TEMP VDD pmos_1p8 L=0.18U W=1.2U
M3 OUT_B A VN 0 nmos_1p8 L=0.18U W=0.3U
M4 OUT_B B VN 0 nmos_1p8 L=0.18U W=0.3U
X1 OUT_B OUT VN VP INVERTER
.ENDS
```

```
*****NEW HALF ADDER with SELECT_BAR, SELECT, and PULL DOWN*****
.SUBCKT HA_NEW SBAR SELECT A B OUT CTLB
M1 VP SBAR VDD VDD pmos_1p8 L=0.18U W=0.9U
M2 VN CTLB TEMP 0 nmos_1p8 L=0.18U W=0.9U
M3 TEMP SELECT 0 0 nmos_1p8 L=0.18U W=0.9U
X1 A B OUT VN VP ORS
M4 OUT CTLB VP VDD pmos_1p8 L=0.18U W=0.9U
M5 OUT SBAR 0 0 nmos_1p8 L=0.18U W=0.9U
.ENDS
```

```
*****28-TRANSISTORS FULL ADDER with SELECT, SELECT_BAR, and PULL DOWN *****
.SUBCKT FA SELECT SELECT_BAR A B CIN SUM COUT
M29 VP SELECT VDD VDD pmos_1p8 L=0.18U W=2.4U
M30 VN SELECT_BAR 0 0 nmos_1p8 L=0.18U W=1.2U
M31 SUM SELECT 0 0 nmos_1p8 L=0.18U W=1.2U
M1 T1 A VP VDD pmos_1p8 L=0.18U W=0.6U
M2 T1 B VP VDD pmos_1p8 L=0.18U W=0.6U
M3 T2 B T1 VDD pmos_1p8 L=0.18U W=0.6U
M4 X A T2 VDD pmos_1p8 L=0.18U W=0.6U
M5 X CIN T1 VDD pmos_1p8 L=0.18U W=0.6U
M6 X CIN T3 0 nmos_1p8 L=0.18U W=0.3U
M7 T3 A VN 0 nmos_1p8 L=0.18U W=0.3U
M8 T3 B VN 0 nmos_1p8 L=0.18U W=0.3U
M9 X A T4 0 nmos_1p8 L=0.18U W=0.3U
M10 T4 B VN 0 nmos_1p8 L=0.18U W=0.3U
M11 T5 CIN VP VDD pmos_1p8 L=0.18U W=0.6U
M12 T5 A VP VDD pmos_1p8 L=0.18U W=0.6U
M13 T5 B VP VDD pmos_1p8 L=0.18U W=0.6U
M14 T6 A T5 VDD pmos_1p8 L=0.18U W=0.6U
M15 T7 B T6 VDD pmos_1p8 L=0.18U W=0.6U
```

```

M16 T8 CIN T7 VDD pmos_1p8 L=0.18U W=0.6U
M17 T8 X T5 VDD pmos_1p8 L=0.18U W=0.6U
M18 T8 X T9 0 nmos_1p8 L=0.18U W=0.3U
M19 T9 A VN 0 nmos_1p8 L=0.18U W=0.3U
M20 T9 B VN 0 nmos_1p8 L=0.18U W=0.3U
M21 T9 CIN VN 0 nmos_1p8 L=0.18U W=0.3U
M22 T8 CIN T10 0 nmos_1p8 L=0.18U W=0.3U
M23 T10 A T11 0 nmos_1p8 L=0.18U W=0.3U
M24 T11 B VN 0 nmos_1p8 L=0.18U W=0.3U
M25 SUM T8 VP VDD pmos_1p8 L=0.18U W=0.6U
M26 SUM T8 VN 0 nmos_1p8 L=0.18U W=0.3U
M27 COUT X VP VDD pmos_1p8 L=0.18U W=0.6U
M28 COUT X VN 0 nmos_1p8 L=0.18U W=0.3U
.ENDS

```

*****FOUR BITS INACCURATE PART*****

```

.SUBCKT LOWER_PART_4 SBAR SELECT A3 A2 A1 A0 B3 B2 B1 B0 CTL0 CTL1 CTL2 CTL3 S0 S1 S2 S3
XADDER0 SBAR SELECT A0 B0 S0 CTL0 HA_NEW
XADDER1 SBAR SELECT A1 B1 S1 CTL1 HA_NEW
XADDER2 SBAR SELECT A2 B2 S2 CTL2 HA_NEW
XADDER3 SBAR SELECT A3 B3 S3 CTL3 HA_NEW
.ENDS

```

*****FOUR BITS ACCURATE PART*****

```

.SUBCKT HIGHER_PART_4 SELECT SELECT_BAR A31 A30 A29 A28 B31 B30 B29 B28 COUT27 S28 S29 S30
S31 COUT31
X1 SELECT SELECT_BAR A28 B28 COUT27 S28 COUT28 FA
X2 SELECT SELECT_BAR A29 B29 COUT28 S29 COUT29 FA
X3 SELECT SELECT_BAR A30 B30 COUT29 S30 COUT30 FA
X4 SELECT SELECT_BAR A31 B31 COUT30 S31 COUT31 FA
.ENDS

```

*****FOUR BITS CONTROL BLOCK with PULL UP when SELECT signal is "0"*****

```

.SUBCKT CONTROL_4 SELECT CTLIN1 CTLIN2 A3 A2 A1 A0 B3 B2 B1 B0 CTL0 CTL1 CTL2 CTL3
XC55 A3 B3 TEMP3 NAND
XC56 CTLIN1 CTLIN2 TEMP3 CTL3 AND3
M1 CTL3 SELECT VDD VDD pmos_1p8 L=0.18U W=0.9U
XC57 A2 B2 TEMP2 NAND
XC58 CTL3 TEMP2 CTL2 AND
M2 CTL2 SELECT VDD VDD pmos_1p8 L=0.18U W=0.9U
XC59 A1 B1 TEMP1 NAND
XC60 CTL2 TEMP1 CTL1 AND
M3 CTL1 SELECT VDD VDD pmos_1p8 L=0.18U W=0.9U
XC61 A0 B0 TEMP0 NAND
XC62 CTL1 TEMP0 CTL0 AND
M4 CTL0 SELECT VDD VDD pmos_1p8 L=0.18U W=0.9U
.ENDS

```

*****SELECT BLOCK, Select signal select the Half-Adder of the next three block*****

```

.SUBCKT SELECT_BLOCK SelectPre A3 A2 A1 A0 B3 B2 B1 B0 Select SBar
M1 temp1 A3 VDD VDD pmos_1p8 L=0.18U W=5.4U
M2 temp2 A2 temp1 VDD pmos_1p8 L=0.18U W=5.4U
M3 temp3 A1 temp2 VDD pmos_1p8 L=0.18U W=5.4U

```

```
M4 temp4 A0 temp3 VDD pmos_1p8 L=0.18U W=5.4U
M5 temp5 B3 temp4 VDD pmos_1p8 L=0.18U W=5.4U
M6 temp6 B2 temp5 VDD pmos_1p8 L=0.18U W=5.4U
M7 temp7 B1 temp6 VDD pmos_1p8 L=0.18U W=5.4U
M8 temp8 B0 temp7 VDD pmos_1p8 L=0.18U W=5.4U
M9 SBar SelectPre temp8 VDD pmos_1p8 L=0.18U W=5.4U
M10 SBar A3 0 0 nmos_1p8 L=0.18U W=0.3U
M11 SBar A2 0 0 nmos_1p8 L=0.18U W=0.3U
M12 SBar A1 0 0 nmos_1p8 L=0.18U W=0.3U
M13 SBar A0 0 0 nmos_1p8 L=0.18U W=0.3U
M14 SBar B3 0 0 nmos_1p8 L=0.18U W=0.3U
M15 SBar B2 0 0 nmos_1p8 L=0.18U W=0.3U
M16 SBar B1 0 0 nmos_1p8 L=0.18U W=0.3U
M17 SBar B0 0 0 nmos_1p8 L=0.18U W=0.3U
M18 SBar SelectPre 0 0 nmos_1p8 L=0.18U W=0.3U
X1 SBar Select 0 VDD INVERTER
.ENDS
```

*****Two input MUX, when select is "1", IN1 is selected*****

```
.SUBCKT MUX2 IN0 IN1 SELECT SELECT_BAR OUT
M1 OUT SELECT_BAR IN0 0 nmos_1p8 L=0.18U W=0.3U
M2 OUT SELECT IN0 VDD pmos_1p8 L=0.18U W=0.6U
M3 OUT SELECT IN1 0 nmos_1p8 L=0.18U W=0.3U
M4 OUT SELECT_BAR IN1 VDD pmos_1p8 L=0.18U W=0.6U
.ENDS
```

*****GETSUM CICUIT*****

```
.SUBCKT GETSUM_4 S3F S2F S1F S0F S3N S2N S1N S0N SELECT SELECT_BAR S3 S2 S1 S0
X1 S3F S3N SELECT SELECT_BAR S3 MUX2
X2 S2F S2N SELECT SELECT_BAR S2 MUX2
X3 S1F S1N SELECT SELECT_BAR S1 MUX2
X4 S0F S0N SELECT SELECT_BAR S0 MUX2
.ENDS
```

*****MAIN CIRCUIT*****

```
X1 SELECT3Bar SELECT3 A3 A2 A1 A0 B3 B2 B1 B0 CTL0 CTL1 CTL2 CTL3 S0N S1N S2N S3N LOWER_PART_4
X2 SELECT7Bar SELECT7 A7 A6 A5 A4 B7 B6 B5 B4 CTL4 CTL5 CTL6 CTL7 S4N S5N S6N S7N LOWER_PART_4
X3 SELECT11Bar SELECT11 A11 A10 A9 A8 B11 B10 B9 B8 CTL8 CTL9 CTL10 CTL11 S8N S9N S10N S11N
LOWER_PART_4
X4 SELECT15Bar SELECT15 A15 A14 A13 A12 B15 B14 B13 B12 CTL12 CTL13 CTL14 CTL15 S12N S13N S14N
S15N LOWER_PART_4
X5 SELECT19Bar SELECT19 A19 A18 A17 A16 B19 B18 B17 B16 CTL16 CTL17 CTL18 CTL19 S16N S17N S18N
S19N LOWER_PART_4

X6 SELECT3 SELECT3Bar A3 A2 A1 A0 B3 B2 B1 B0 0 S0F S1F S2F S3F COUT3 HIGHER_PART_4
X7 SELECT7 SELECT7Bar A7 A6 A5 A4 B7 B6 B5 B4 COUT3 S4F S5F S6F S7F COUT7 HIGHER_PART_4
X8 SELECT11 SELECT11Bar A11 A10 A9 A8 B11 B10 B9 B8 COUT7 S8F S9F S10F S11F COUT11
HIGHER_PART_4
X9 SELECT15 SELECT15Bar A15 A14 A13 A12 B15 B14 B13 B12 COUT11 S12F S13F S14F S15F COUT15
HIGHER_PART_4
X10 SELECT19 SELECT19Bar A19 A18 A17 A16 B19 B18 B17 B16 COUT15 S16F S17F S18F S19F COUT19
HIGHER_PART_4
```

```

X11 0 VDD A23 A22 A21 A20 B23 B22 B21 B20 COUT19 S20 S21 S22 S23 COUT23 HIGHER_PART_4
X12 0 VDD A27 A26 A25 A24 B27 B26 B25 B24 COUT23 S24 S25 S26 S27 COUT27 HIGHER_PART_4
X13 0 VDD A31 A30 A29 A28 B31 B30 B29 B28 COUT27 S28 S29 S30 S31 COUT31 HIGHER_PART_4

X14 SELECT19 VDD VDD A19 A18 A17 A16 B19 B18 B17 B16 CTL16 CTL17 CTL18 CTL19 CONTROL_4
X15 SELECT15 CTL19 CTL16 A15 A14 A13 A12 B15 B14 B13 B12 CTL12 CTL13 CTL14 CTL15 CONTROL_4
X16 SELECT11 CTL15 CTL12 A11 A10 A9 A8 B11 B10 B9 B8 CTL8 CTL9 CTL10 CTL11 CONTROL_4
X17 SELECT7 CTL11 CTL8 A7 A6 A5 A4 B7 B6 B5 B4 CTL4 CTL5 CTL6 CTL7 CONTROL_4
X18 SELECT3 CTL7 CTL4 A3 A2 A1 A0 B3 B2 B1 B0 CTL0 CTL1 CTL2 CTL3 CONTROL_4

X19 0 A31 A30 A29 A28 B31 B30 B29 B28 SELECT19 SELECT19Bar SELECT_BLOCK
X20 SELECT19 A27 A26 A25 A24 B27 B26 B25 B24 SELECT15 SELECT15Bar SELECT_BLOCK
X21 SELECT15 A23 A22 A21 A20 B23 B22 B21 B20 SELECT11 SELECT11Bar SELECT_BLOCK
X22 SELECT11 A19 A18 A17 A16 B19 B18 B17 B16 SELECT7 SELECT7Bar SELECT_BLOCK
X23 SELECT7 A15 A14 A13 A12 B15 B14 B13 B12 SELECT3 SELECT3Bar SELECT_BLOCK

X24 S3F S2F S1F S0F S3N S2N S1N S0N SELECT3 SELECT3Bar S3 S2 S1 S0 GETSUM_4
X25 S7F S6F S5F S4F S7N S6N S5N S4N SELECT7 SELECT7Bar S7 S6 S5 S4 GETSUM_4
X26 S11F S10F S9F S8F S11N S10N S9N S8N SELECT11 SELECT11Bar S11 S10 S9 S8 GETSUM_4
X27 S15F S14F S13F S12F S15N S14N S13N S12N SELECT15 SELECT15Bar S15 S14 S13 S12 GETSUM_4
X28 S19F S18F S17F S16F S19N S18N S17N S16N SELECT19 SELECT19Bar S19 S18 S17 S16 GETSUM_4

.MEASURE TRAN TOTAL_POWER AVG POWER FROM=0 TO=1u

.MEAS TRAN Tdelay TRIG V(A0) val=0.9 TD=0n RISE=1 TARG V(S12) val=0.9 RISE=1
*.MEAS TRAN Tdelay TRIG V(A24) val=0.9 TD=0n RISE=1 TARG V(S31) val=0.9 FALL=1

.LIB 'D:\WeiJia\hspice\HSpice_Models\sm093001-1j.hspice' typical
.END

```

Appendix D – HSPICE Code of ETAIV

ETAIV

*****NOTES*****

* This is an implementation of the 32-bit ETA *

* Author: Zhu Ning

```
.OPTIONS NODE POST                                $The option POST generates AvanWaves files for
comparing the model to the data;
.VEC 'E:\phd file\Wang Gang\My design\Data\input_adder_32.dat'
.GLOBAL VDD
.PARAM NPL=0.18u NW=0.3u PW=0.6u VSUPPLY=1.8V      $VDD=1.8V
VDD VDD GND VSUPPLY
.TRAN 1n 1u                                         $Perform and print the transient analysis every 1ns for 1us;
*.TRAN 0.1n 20n                                   $Another transient analysis;
```

*****INVERTER*****

*2 transistors totally

.SUBCKT INVERTER IN OUT

M1 OUT IN VDD VDD pmos_1p8 L=NPL W=PW

M2 OUT IN GND GND nmos_1p8 L=NPL W=NW

.ENDS

*****2-input AND gate*****

*6 transistors totally

.SUBCKT AND2 IN1 IN2 OUT

M1 SUBOUT IN1 VDD VDD pmos_1p8 L=NPL W=PW

M2 SUBOUT IN2 VDD VDD pmos_1p8 L=NPL W=PW

M3 SUBOUT IN1 COM GND nmos_1p8 L=NPL W=0.6u

M4 COM IN2 GND GND nmos_1p8 L=NPL W=0.6u

\$W=2NW to decrease the delay at two

HIGH inputs;

X1 SUBOUT OUT INVERTER

\$call for inverter;

.ENDS

*****2-input OR gate*****

*6 transistor totally

.SUBCKT OR2 IN1 IN2 OUT

M1 COM IN1 VDD VDD pmos_1p8 L=NPL W=1.2u

M2 SUBOUT IN2 COM VDD VDD pmos_1p8 L=NPL W=1.2u

\$W=2PW;

M3 SUBOUT IN1 GND GND nmos_1p8 L=NPL W=NW

M4 SUBOUT IN2 GND GND nmos_1p8 L=NPL W=NW

X1 SUBOUT OUT INVERTER

.ENDS

*****2-TO-1 MUX using transmission gate*****

*OUT=A if SELECT=VDD

*OUT=B if SELECT=0

.SUBCKT MUX SELECT- A B OUT

M1 OUT SELECT- A VDD pmos_1p8 L=NPL W=PW

```

M2 A SELECT OUT GND nmos_1p8 L=NPL W=NW
M3 OUT SELECT B VDD pmos_1p8 L=NPL W=PW
M4 B SELECT- OUT GND nmos_1p8 L=NPL W=NW
.ENDS
*****

*****FOUR-BIT CLA LOGIC*****

.SUBCKT CLA_LOGIC CIN G3 G2 G1 G0 P3 P2 P1 P0 COUT COUT-
M1 PD1 G3 VDD VDD pmos_1p8 L=NPL W=PW
M2 PD2 G2 PD1 VDD pmos_1p8 L=NPL W=PW
M3 PD3 G1 PD2 VDD pmos_1p8 L=NPL W=PW
M4 PD4 G0 PD3 VDD pmos_1p8 L=NPL W=PW
M5 COUT- CIN PD4 VDD pmos_1p8 L=NPL W=PW
M6 COUT- P3 VDD VDD pmos_1p8 L=NPL W=PW
M7 COUT- P2 PD1 VDD pmos_1p8 L=NPL W=PW
M8 COUT- P1 PD2 VDD pmos_1p8 L=NPL W=PW
M9 COUT- P0 PD3 VDD pmos_1p8 L=NPL W=PW
M10 COUT- CIN ND4 GND nmos_1p8 L=NPL W=NW
M11 ND4 P0 ND3 GND nmos_1p8 L=NPL W=NW
M12 ND3 P1 ND2 GND nmos_1p8 L=NPL W=NW
M13 ND2 P2 ND1 GND nmos_1p8 L=NPL W=NW
M14 ND1 P3 GND GND nmos_1p8 L=NPL W=NW
M15 COUT- G0 ND3 GND nmos_1p8 L=NPL W=NW
M16 COUT- G1 ND2 GND nmos_1p8 L=NPL W=NW
M17 COUT- G2 ND1 GND nmos_1p8 L=NPL W=NW
M18 COUT- G3 GND GND nmos_1p8 L=NPL W=NW
X1 COUT- COUT INVERTER
.ENDS
*****

*****P&G GENERATOR*****

.SUBCKT PG4 A3 A2 A1 A0 B3 B2 B1 B0 G3 G2 G1 G0 P3 P2 P1 P0
X1 A0 B0 G0 AND2
X2 A1 B1 G1 AND2
X3 A2 B2 G2 AND2
X4 A3 B3 G3 AND2
X5 A0 B0 P0 OR2
X6 A1 B1 P1 OR2
X7 A2 B2 P2 OR2
X8 A3 B3 P3 OR2
.ENDS
*****

*****28-TRANSISTORS FULL ADDER*****

.SUBCKT FA A B CIN SUM COUT
M1 T1 A VDD VDD pmos_1p8 L=NPL W=PW
M2 T1 B VDD VDD pmos_1p8 L=NPL W=PW
M3 T2 B T1 VDD pmos_1p8 L=NPL W=PW
M4 X A T2 VDD pmos_1p8 L=NPL W=PW
M5 X CIN T1 VDD pmos_1p8 L=NPL W=PW
M6 X CIN T3 0 nmos_1p8 L=NPL W=NW
M7 T3 A 0 0 nmos_1p8 L=NPL W=NW
M8 T3 B 0 0 nmos_1p8 L=NPL W=NW

```

```
M9 X A T4 0 nmos_1p8 L=NPL W=NW
M10 T4 B 0 0 nmos_1p8 L=NPL W=NW
M11 T5 CIN VDD VDD pmos_1p8 L=NPL W=PW
M12 T5 A VDD VDD pmos_1p8 L=NPL W=PW
M13 T5 B VDD VDD pmos_1p8 L=NPL W=PW
M14 T6 A T5 VDD pmos_1p8 L=NPL W=PW
M15 T7 B T6 VDD pmos_1p8 L=NPL W=PW
M16 T8 CIN T7 VDD pmos_1p8 L=NPL W=PW
M17 T8 X T5 VDD pmos_1p8 L=NPL W=PW
M18 T8 X T9 0 nmos_1p8 L=NPL W=NW
M19 T9 A 0 0 nmos_1p8 L=NPL W=NW
M20 T9 B 0 0 nmos_1p8 L=NPL W=NW
M21 T9 CIN 0 0 nmos_1p8 L=NPL W=NW
M22 T8 CIN T10 0 nmos_1p8 L=NPL W=NW
M23 T10 A T11 0 nmos_1p8 L=NPL W=NW
M24 T11 B 0 0 nmos_1p8 L=NPL W=NW
M25 SUM T8 VDD VDD pmos_1p8 L=NPL W=PW
M26 SUM T8 0 0 nmos_1p8 L=NPL W=NW
M27 COUT X VDD VDD pmos_1p8 L=NPL W=PW
M28 COUT X 0 0 nmos_1p8 L=NPL W=NW
.ENDS
```

*****4-BIT RIPPLE CARRY ADDER*****

```
.SUBCKT RCA4 C0 A3 A2 A1 A0 B3 B2 B1 B0 S3 S2 S1 S0 C4
X1 A0 B0 C0 S0 C1 FA
X2 A1 B1 C1 S1 C2 FA
X3 A2 B2 C2 S2 C3 FA
X4 A3 B3 C3 S3 C4 FA
.ENDS
```

*****8-BIT SUM BLK*****

```
.SUBCKT ADDER8 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
COUT12 S11 S10 S9 S8 S7 S6 S5 S4
X1 A3 A2 A1 A0 B3 B2 B1 B0 G3 G2 G1 G0 P3 P2 P1 P0 PG4
X2 0 G3 G2 G1 G0 P3 P2 P1 P0 C4 C4- CLA_LOGIC
X3 A7 A6 A5 A4 B7 B6 B5 B4 G7 G6 G5 G4 P7 P6 P5 P4 PG4
X4 0 G7 G6 G5 G4 P7 P6 P5 P4 C8_0 C8_0- CLA_LOGIC
X5 VDD G7 G6 G5 G4 P7 P6 P5 P4 C8_1 C8_1- CLA_LOGIC
X6 C4 C4- C8_1 C8_0 C8 MUX $SELECT THE CARRY OUTPUT FOR NEXT 4-BIT BLK
X7 C4 A7 A6 A5 A4 B7 B6 B5 B4 S7 S6 S5 S4 COUT8 RCA4
X8 C8 A11 A10 A9 A8 B11 B10 B9 B8 S11 S10 S9 S8 COUT12 RCA4
.ENDS
```

*****MAIN CIRCUIT*****

```
X1 A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21
B20 COUT32 S31 S30 S29 S28 S27 S26 S25 S24 ADDER8
X2 A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 B23 B22 B21 B20 B19 B18 B17 B16 B15 B14 B13
B12 COUT24 S23 S22 S21 S20 S19 S18 S17 S16 ADDER8
X3 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 COUT16
S15 S14 S13 S12 S11 S10 S9 S8 ADDER8
```

```
X4 A3 A2 A1 A0 B3 B2 B1 B0 G3 G2 G1 G0 P3 P2 P1 P0 PG4
X5 0 G3 G2 G1 G0 P3 P2 P1 P0 C4 C4- CLA_LOGIC
X6 C4 A7 A6 A5 A4 B7 B6 B5 B4 S7 S6 S5 S4 COUT8 RCA4
X7 0 A3 A2 A1 A0 B3 B2 B1 B0 S3 S2 S1 S0 COUT4 RCA4
```

```
.MEASURE TRAN TOTAL_POWER AVG POWER FROM=0 TO=1u
```

```
.MEAS TRAN Tdelay TRIG V(A20) val=0.9 TD=0n RISE=1 TARG V(COUT32) val=0.9 RISE=1
```

```
.LIB 'E:\phd file\Wang Gang\My design\HSpice_Models\sm093001-1j.hspice' typical
.END
```


Appendix E – MATLAB code of FFT

```
%The fourier transform implementation with ETAllI
%Author: Zhu Ning
%Created: 1 Sep 2009
%Last Modified: 1 Sep 2009
function output = FourierTransformExp(input);

N1=2^32; %total bit
N2=2^20; %inaccurat part bit
N2Reset = 2^20;

output = input;
output_real = input;
output_image = input;

N = length(input);
%find N
%l=log2(leng);
%p=ceil(l);
%N = 2^p;
%display(N);

c = -i*2*pi/N; %constant value for exp
%display(c);

for k=1:N
    output(k) = 0;
    output_real(k) = 0;
    output_image(k) = 0;

    for j=1:N
        temp = input(j)*exp(c*k*j);
        %output(k) = output(k) + temp;

        %replace the addtion with ETAllI
        temp_real = real(temp);
        temp_image = imag(temp);

        %calculate N2 for ETAllI of real part
        N2 = N2Reset; %reset
        while (output_real(k)<N2 && temp_real<N2 && N2>1)
            N2 = N2 / 16; %shift 4-bits
        end
        %do the addition
        if (N2 == 1)
            output_real(k) = output_real(k) + temp_real; %normal addition
        else
            output_real(k) = addition(output_real(k), temp_real, N1, N2);
        end

        %calculate N2 for ETAllI of image part
        N2 = N2Reset; %reset
```

```
while (output_image(k)<N2 && temp_image<N2 && N2>1)
    N2 = N2 / 16; %shift 4-bits
end
%do the addition
if (N2 == 1)
    output_image(k) = output_image(k) + temp_image; %normal addition
else
    output_image(k) = addition(output_image(k), temp_image, N1, N2);
end
end
output(k) = complex(output_real(k), output_image(k));
%display(output(k));
end
```

Appendix F – MATLAB code of ETAlII addition algorithm

```
function result=addition(x,y,N1,N2)
```

```
%N1=16;  
%N2=4;  
z=0;  
x1=0;  
y1=0;  
x2=0;  
y2=0;  
z1=0;  
z2=0;  
n=N2;  
flag=0;  
  
x1=floor(x/N2);  
y1=floor(y/N2);  
z1=(x1+y1)*N2;  
x2=x-x1*N2;  
y2=y-y1*N2;  
n=n/2;  
while(flag==0 && n>0)  
    xt=floor(x2/n);  
    yt=floor(y2/n);  
    if((xt+yt)==1)  
        z2=z2+n;  
    else  
        if(xt==1 && yt==1)  
            z2=z2+n*2-1;  
            flag=1;  
        end  
    end  
    x2=x2-xt*n;  
    y2=y2-yt*n;  
    n=floor(n/2);  
end  
z=z1+z2;  
result=z;
```